Software Documentation

of the

Photon Conversion Group

Table of Contents

Introduction

AliPhysics Implementation and GRID Running

Quality Assurance and Energy Calibration of Calorimeters

Cocktail Running and External Input

Neutral Meson and Direct Photon Analysis - Afterburners

Welcome to the Photon Conversion Group Tutorial

This tutorial is meant as an introduction to the software packages provided by the Photon Conversion Group (PCG). As it is still under development as is the software it does not claim completeness. However, it should provide a good start for new-comers and oldies to get started on a new topic. If you wanna help improving this documentation you are very welcome and just need to let us know and e-mail Friederike. She will grant you access to the documentation git as well.

In the provided software package the reconstruction of photons within ALICE using different detectors (EMCal, DCal, PHOS, TPC & ITS) is handled in a common way and the neutral mesons can be extracted using different techniques. Furthermore the software can be used as analysis level QA for the corresponding detectors focussing on the quantities relevant for the corresponding photon reconstruction.

Our software is structured in 3 main parts hosted on GitHub and GitLab:

1. **AliPhysics** implementation in **PWGGA/GammaConv**, which is running on the reconstructed ESD/AOD data or simulation files
2. **Afterburner** implementation which is kept and maintained in the **PCG-Software directory** and is used to analyse the output files of the AliPhysics-Tasks to extract the meson spectra (π^0, η, ω), v_n and direct photon results. It is named *AnalysisSoftware* repository
3. **Cocktail** implementation, which is scattered in 2 different places in AliPhysics (**PWG/Cocktail** & **PWGGA/GammaConv**) and which has to fed by inputs summarized in the **Cocktail-Repository**

All three packages will be needed to complete the full meson and direct photon analysis as it is currently implemented and will be explained in the following tutorial in different sections.

Furthermore, we have a repository (named AnalysisNotes) hosting all analysis/combination notes **PCG-Notes repository** where you should contribute with your own analysis note once your analysis reached an advanced stage and is targeted to be included in a publication.

The *AliPhysics* repository is automatically installed, the rest of the repositores can be cloned to your local disk (remember the SSH key registration!) via the following commands:

- AnalysisSoftware

  ```
  git clone ssh://git@gitlab.cern.ch:7999/alice-pcg/AnalysisSoftware.git
  ```

- AnalysisNotes

  ```
  git clone ssh://git@gitlab.cern.ch:7999/alice-pcg/AnalysisNotes.git
  ```

- Cocktail

  ```
  git clone ssh://git@gitlab.cern.ch:7999/alice-cocktail-EM/cocktail_input.git
  ```

To gain committer-access to the PCG-Software directories (AnalysisSoftware and AnalysisNotes) please subscribe to ***alice-pcg*** **&** ***alice-pcg-core*** on e-groups and to get the committer access to the Cocktail repository you have to be subscribed to ***alice-cocktail-EM***. For AliPhysics such a subscribtion is not needed as you create your own fork of the repository and do pull-requests as described in Working with AliPhysics.

Last but not least, the last GitHub repository is our GitBook tutorial repository to complete the list (which you are reading) **PCG-tutorial** which is linked to the GitBook itself **Tutorial**

Contacts

If there are any questions regarding this tutorial do not hesitate to ask on

- alice-pcg-core@cern.ch

or directly to the convenors and creators of the tutorial

- Daniel Muehlheim d.muehlheim@cern.ch
- Friederike Bock friederike.bock@cern.ch
- Nicolas Schmidt nicolas.schmidt@cern.ch
- Mike Sas msas@nikhef.nl

We are happy to help.

Communication Tools

Besides the egroup and email contact, the *PCG* uses a Slack group PCM @ Slack. If you want to join the group, you need to have an invitation. Please write an e-mail to one of the creators of the tutorial for this purpose.

Weekly meetings of the PCG

Weekly meetings of the PCG take place on vidyo PCG @ Indico. They are usually scheduled for Thursday, 10am, and open to anybody with interest in the topic. Announcements are sent around via the main egroup

- alice-pcg@cern.ch

which you should be registered to in addition to alice-pcg-core@cern.ch -> see e-groups

Additional Information

useful links

- AliPhysics, aliBuild, AliEn

twikis:

- PCG
- PWGGA
- EMCocktail

Git

Git is a very powerful version control system which is very popular (and it is also used for all our software repositories). You are expected to know about Git and to be able to work with it. If you don't know how to push, pull, rebase and configure Git, you should have a look at the given links below and learn about Git.

It is recommended to use a visualization tool as it eases your workflow and gives you direct insight into the state of a repository and the status of all the branches. There are two different tools which are recommended to use:

1: *gitk*

In general, you can install it via your package manager, for example:

```
sudo apt-get install gitk
```

For more information see gitk.

2: *tig*

In general, you can install it via your package manager, for example:

```
sudo apt-get install tig
```

For more information see tig.

Suggested workflow

Suggested workflow

Basic GIT workflow in our repositories

Checkout your development branch in the given repository

```
git checkout UserBranch
```

make your changes to the macros and/or add new files (i.e. ExtractSignalBinning.h)

```
git add ExtractSignalBinning.h
```

Write a commit message and make the commit itself via:

```
git commit -m "Changed binning for some energy"
```

Checkout the master branch and rebase to the latest version

```
git checkout master
git pull --rebase
```

Checkout your branch again and rebase it to the latest master version

```
git checkout UserBranch
git rebase master
```

If everything works with the rebase, then push the changes.

```
git push origin UserDevel
```

Afterwards, a merge request can be made on the git repositories website.

Useful Links

- General Git tutorial
- ALICE Git tutorial

General Naming Scheme and Analysis Task Names

In our framework we follow a common naming scheme and several abbreviations will occur frequently. When we are talking about the **meson analysis**, we usually refer to the *neutal pion and eta analysis*. While when we are talking about the **photon analysis** we generally mean the *direct photon analysis*.
Within the software package different photon reconstruction techniques are implemented:

Abbreviation	Explanation
Conversions (Conv, PCM)	Photon reconstruction via conversion in the detector material based on the electrons and positrons reconstructed in the TPC
Calo (EMCal, DCal PHOS)	Photons reconstructed via their energy deposit in the calorimeters

These are then combined to reconstruct the neutral mesons in either the *two photon* ($\gamma\gamma$) or the *Dalitz decay channel* (γe^+e^-) using the same or different techniques or detectors for the photon reconstruction. The corresponding analysis tasks in **AliPhysics** in **PWGGA/GammaConv** are:

- AliAnalysisTaskGammaConvV1.cxx
- AliAnalysisTaskGammaCalo.cxx
- AliAnalysisTaskGammaConvCalo.cxx
- AliAnalysisTaskGammaConvDalitzV1.cxx
- AliAnalysisTaskGammaCaloDalitzV1.cxx
- AliAnalysisTaskGammaCaloMerged.cxx

These tasks are named according to their photon/meson reconstruction method and will in most cases be used to reconstruct not only the π^0 and η meson but also the direct photons. A similar naming scheme will be used throughout the whole software package and is explained in the following.

Abbreviation	Explanation
Conv, PCM	Photons reconstructed using conversions and then paired with other photons reconstructed with the same technique
Calo	Photons reconstructed using one of the three calorimeters (EMCal, DCal, PHOS) and then paired with other photons reconstructed with the same technique
ConvCalo	Photons reconstructed using one of the three calorimeters (EMCal, DCal, PHOS) and then paired with photons, which have been reconstructed from conversions. Currently 3 combinations are possible: PCM-EMC, PCM-DMC, PCM-PHOS. For the direct photon reconstruction the conversion photon is used for the inclusive photon reconstruction.
ConvDalitz	Photons reconstructed using conversions and then paired with 2 primary electrons reconstructed in the TPC and ITS.
CaloDalitz	Photons reconstructed using one of the three calorimeters (EMCal, DCal, PHOS) and then paired with 2 primary electrons reconstructed in the TPC and ITS.
CaloMerged, Merged	Neutral pions and eta mesons are reconstructed in the calorimeter via single clusters in the region where the 2 photons can no longer be separated due to the calorimeter resolution. These techniques are also refered to as mEMC, mDMC or mPHOS, where only the first has been explored so far.

They are configure by the AddTasks in **PWGGA/GammaConv/macros** following a similar naming scheme:

AddTask_Gamma[Calo,ConvV1,ConvCalo,ConvDalitzV1,CaloDalitzV1,CaloMerged]_[pp,pPb,PbPb].C

These tasks produce output files which are named according to the reconstruction method and the train-

configuration which has been chosen i.e.: *GammaConvV1*\$TRAINCONFIG.root, GammaCalo_$TRAINCONFIG.root, GammaConvCalo_$TRAINCONFIG.root, GammaCaloDalitz_$TRAINCONFIG.root, GammaConvDalitzV1_$TRAINCONFIG.root, GammaCaloMerged_$TRAINCONFIG.root_

For the heavier meson reconstruction using the π^0 in the decay chain the same convention for the names could not be kept and the corresponding tasks are named according to the decay chain and meson which they are supposed to reconstruct:

- AliAnalysisTaskK0toPi0Pi0.cxx
- AliAnalysisTaskOmegaToPiZeroGamma.cxx
- AliAnalysisTaskNeutralMesonToPiPlPiMiPiZero.cxx
- AliAnalysisTaskEtaToPiPlPiMiGamma.cxx

In these cases the photon/neutral pion reconstruction technique is selected through the configurations in the AddTask, which follow a similar naming scheme as explained before.
There are several "helper-tasks" implemented in **PWGGA/GammaConv** in addition, which are usually used for QA purposes, Cocktail or MC investigations. Furthermore they can be used to determine the material budget uncertainty.

- AliAnalysisTaskConversionQA.cxx
- AliAnalysisTaskGammaCocktailMC.cxx
- AliAnalysisTaskHadronicCocktailMC.cxx
- AliAnalysisTaskGammaPureMC.cxx
- AliAnalysisTaskResolution.cxx
- AliAnalysisTaskMaterial.cxx
- AliAnalysisTaskMaterialHistos.cxx

For the conversion analysis also the v_n reconstruction for the photons and pions is implemented in **PWGGA/GammaConv**, which is not yet the case for the corresponding calorimeter based analysis.

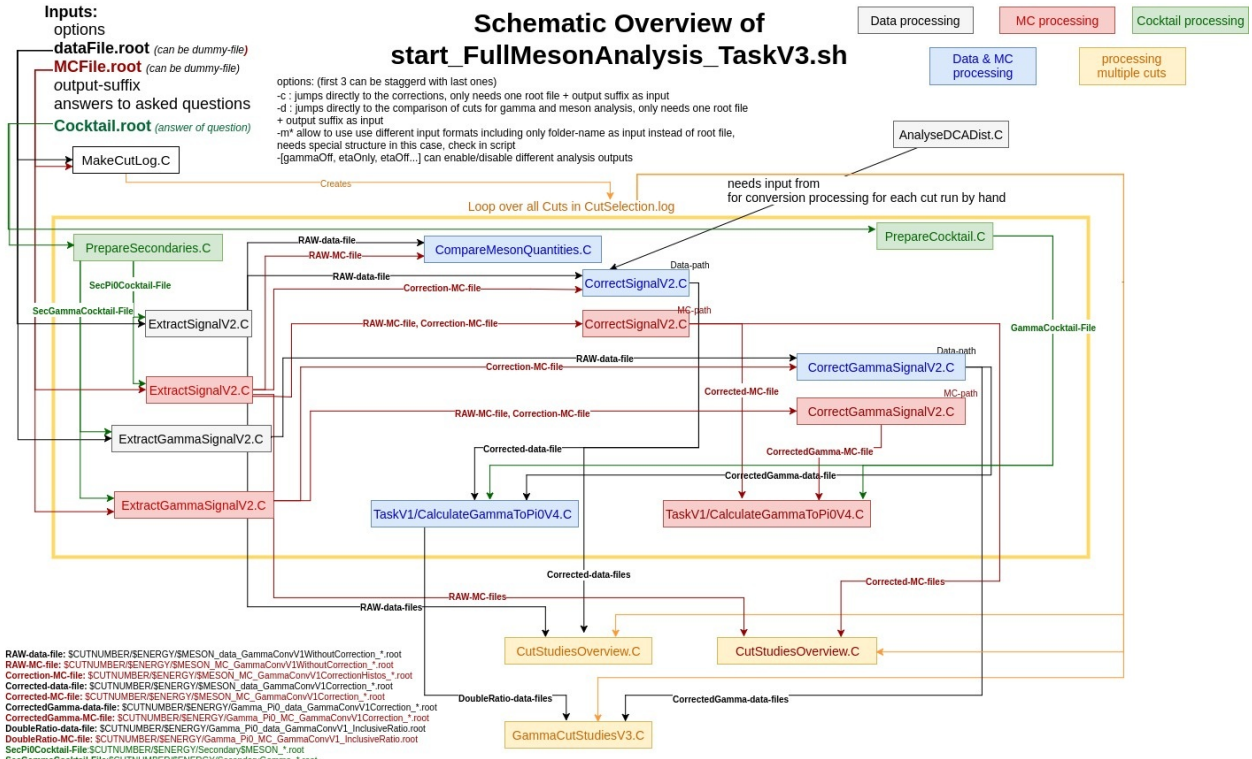
- AliAnalysisTaskGammaConvFlow.cxx
- AliAnalysisTaskPi0v2.cxx

General Afterburner Introduction

The analysis of the neutral mesons and direct photons, which is done after the pure grid analysis is organized in different macros, where we tried to have one macro per analysis step. Thus we split the steps of signal extraction, out-of-bunch pileup calculation, correction, cocktail processing, double ratio calculation, cut comparsion and systematics calculation into different macros. However, as the most of the macros need input from a previous step and it would be cumbersome to change the macro calls each time with the appropriate file. Most of steps, which need no manual intervention or optimization are handled by our steering scripts:

- ***start_FullMesonAnalysis_TaskV3.sh*** - handling the π^0 & η in the $\gamma\gamma$-decay channel and the merged analysis as well as direct photon analysis
- ***start_FullMesonAnalysisDalitz_TaskV2.sh*** - handling the π^0 & η in the γe^+e--decay channel
- ***start_FullOmegaMesonAnalysis.sh*** - handling the η and ω analysis in the $\pi^0\pi^+\pi^-$ and $\pi^0\gamma$ channel

They are stored in the main directory of the PCG-Afterburner directory. A visualisation of the paths for the neutral meson analysis in the $\gamma\gamma$-channel and the direct photon analysis can be found below using the *start_FullMesonAnalysis_TaskV3.sh*. It explains the main path of the analysis in the standard setup.



However, as usual there are some exceptions and you might want to read the corresponding script ones if you are analysing a well know data set for the first time, as for instance it might be necessary to run a special merging step for the efficiencies from added signals and minimum bias or jet-jet simulations. Each of the three scripts is also equipped with a short helper page, which can be called '--help' or '-h' after the script name.

In general the repository is ordered as follows:

- **main directory**:
  - Contains general steering scripts for the macros (*start_FullMesonAnalysis_TaskV3.sh, start_FullMesonAnalysisDalitz_TaskV2.sh, start_FullOmegaMesonAnalysis*), as mentioned above.
  - Furthermore, the macros to calculate the systematics of the different analysis techniques

(*FinaliseSystematicErrors[Calo,Conv,ConvCalo,Merged,Dalitz]*\[$SYSTEM$ENERGY].C), *the combination macros for the mesons/gammas from different reconstruction techniques* (CombineMesonMeasurements_*.C, CombineGammaResults_*.C) as well as comparison macros among different energies and particles (*CombineNeutralPion*.C, CompareCharged*.C, CompareGamma*.C*). Most of these are tailored to specific energies and cannot be used as generalized macros, but have either been used for the corresponding publications or for the preliminary creation.

- Additional import macros contained in the main directory regarding the publications are the *CalculateReference.C*, *CalculateSignificanceToPYTHIA.C, ComputeCorrelationFactors.C, TestMtScaling.C* which can be used in a slightly more general way. In addition to these there are more specific macros, which were used for the publications (p-Pb mesons and the first meson paper), which are not necessarily maintained any longer.
- The compilation macros of the external inputs for different data files from ALICE and other experiments (*PrepareChargedPionDataALICE*.C,ProduceExperimentalDataGraphsPbPb.C*) and theory predictions (*ProduceTheoryGraphs*.C*) can also be found here.
- In addition to these the *CompileCorrectGammaV2.C* is stored here, which needs to be adjusted to contain your user-name and path to RooUnfold-headers if you want to run the direct photon extraction.
- Last but not least a very helpful shell-script to create soft-links on your computer to one common software directory is stored here *prepareResultsDirectory.sh*. it is recommended to only checkout the PCG-repository once and then create soft-links to that directory in whatever other directory you are running the analysis on different data sets. It can be run by typing:

```
bash $PATHTOPCGGIT/prepareResultsDirectory.sh $USERNAME [pp2760GeV,pPb5TeV,pp7TeV,pp8TeV,pp13TeV]
```

  with the last parameter being optional and the $USERNAME which needs to be added including the path. If the last parameter is used the number of links will be reduced in the main-dir to only include the necessary macros for that specific collision system.

- **CocktailInput**: Contains the output files from the cocktail generation on the grid, which are needed as input for the Afterburners once they are final. Furthermore, contains the final processed files by *PrepareCocktail.C* if they are close to preliminary or publication. We ask everyone not to commit too large root files here, as these will blow up the size of the repository and cannot be deleted fully again.

- **CommonHeaders**: Contains the header files which are commonly used in different macros:

  - *AdjustHistRange.h* contains the functions to automatically adjust the ranges for the QA output included in **TaskQA/**
  - *CombinationFunctions.h* contains all functions (and support routines) to combine results from different detectors/triggers (`CombinePtPointsSpectraFullCorrMat()` , `CombinePtPointsSpectraTriggerCorrMat()`) or compare spectra with different or the same binnings (`CalculateRatioBetweenSpectraWithDifferentBinning()`). Furthermore the previously used functions to combined only the fully independent results from PHOS and PCM are kept in this header files (`CombinePtPointsSpectraAdv()`, `CombinePtPointsSpectra()`, `CombinePtPointsRAA()`), however, they should not be used anymore as they will no longer be maintained. Most of these functions do rather delicate operations (i.e. matrix inversions, transformations of graphs & histos into each other, ...) as such please check the `couts` , they are ment as control and debug output. Last but not least it contains a function to calculate weighted quantities based on the p_T-dependent weights obtained in `CombinePtPointsSpectraTriggerCorrMat()` to properly display for instance the efficiency, acceptance, mass-position... . This function is, however, not yet fully vetted for all special cases, thus please have a look at the outcome and make sure the results are sensible.
  - *ConversionFunctionsBasicsAndLabeling.h* contains all common naming functions depending on the mode, trigger or cutnumber itself. Furthermore, it contains the functions which return the corresponding values of the different cuts and other general quantities, like σ or N_{coll}.
  - *ConversionFunctions.h* contains all general functions used in the Afterburners not regarding plotting,

labeling or combinations. For instance it contains functions to calculate ratios between different graphs, histos, fits and spline as well as scale functions for most of these objects. If you are not sure whether we implemented a more general operation in our framework have a look here or ask in our slack chats. it is very likely one of us already had to deal with the problem you're facing and found a solution.

- *ExtractSignalBinning.h* contains all p_T binnings for the different meson and photon analyses as well as the corresponding functions to set them. If you want to adjust a binning for a certain energy i.e. pp @13TeV make sure it is implemented correctly and does not interfer with anyone elses analysis settings here. Furthermore please make sure it is done in all the proper places, i.e. check on a different energy example what need to be done. We are currently in the process of trying to simplyfy the settings a little, so please implement the newer settings in `InitializeBinning()` using the functions: `InitializeClusterBinning(), GetStartBin(), ReturnSingleInvariantMassBinPlotting(), GetOptimumNColumnsAndRows() & GetBinning()` for the corresponding variables, as these can/will be used in the combination macros for instance as well.
- *ExtractSignalPlotting.h* contains all functions needed for the plotting of the different invariant mass $M_{\gamma\gamma}$, distance of closest approach dca_z or shower shape σ^2_{long} slice in transverse momentum. This can either be done as single invariant mass bin plot or a plot with all slices in one plot. Also the functions for the heavier meson analysis are contained in this header file.
- *FittingGammaConversion.h* contains the definitions of the fitting functions and a generalized function to initialize and fit graphs & histos with predefined functions used for the spectra fitting (`FitObject()`). Also the fitting functions for the resolution fitting are contained in this header.
- *PlottingGammaConversionAdditional.h* This header contains a multitude of functions to make your plots even nicer and let them appear in the common sytle of the conversion/PWGGA group. For instance we defined common colors, marker styles and marker sizes for the different collision systems, energies (`GetColorDefaultColor(), GetDefaultMarkerStyle(), GetDefaultMarkerSize()`), triggers (`GetDefaultTriggerColor(), GetDefaultTriggerMarker(), GetDefaultTriggerColorName(), GetDefaultTriggerMarkerStyleName(), GetDefaultTriggerMarkerSizeName()`) and detection techniques (`GetDefaultColorDiffDetectors, GetDefaultMarkerStyleDiffDetectors(), GetDefaultMarkerSizeDiffDetectors`). Make sure to use them as it eases following presentations in the different meetings and the corresponding colors often have already been optimzed for visibility and style in the corresponding papers. In addition several functions to put logos and labels on the plot are contained in this header (i.e. `PutALICESimulationLabel(), PutProcessLabelAndEnergyOnPlot(), DrawAliceLogoPerformance(), DrawStructure(), DrawAliceText()`). Also some functions to extract the marker settings and colors from histos and graphs and create a single marker or box from these.
- *PlottingGammaConversionHistos.h* contains all functions of the conversion group regarding plotting styles of hists, graphs, fits, profiles as well as some basic functions to obtain the correct text sizes for the different canvases and pads in canvases. The settings can either be done for single objects (i.e. `DrawGammaSetMarker(), DrawGammaNLOTGraph(), SetStyleGammaNLOTGraphWithBand(), DrawAutoGammaHisto(), SetStyleHisto(), SetStyleHistoTH2ForGraphs(), DrawGammaHistoWithTitle2()` ...) or multiple objects (`DrawAutoGammaHistos(), DrawAutoGammaHistosMaterial(), DrawAutoGamma3Histos(), DrawAutoGammaHistosWOLeg()` ...). Most of these functions exist for multiple objects and are named accordingly, they each have slightly different default settings and are usually optimized for a certain purpose. Thus please have a look at the different macros in which context they are used or ask in the slack chat if you are not sure. Additionally, some more general functions for setting proper canvas and pad settings are defined here: `StyleSettings(), StyleSettingsThesis(), SetPlotStyle(), SetPlotStyleNConts(), DrawCanvasSettings(), DrawPadSettings()` using part or all of these functions will make your code much more readable and your plots instantly better visible so please use them. Also the function: `ReturnCorrectValuesForCanvasScaling()` is vary handy as it return the correct values for the pad margins for multipad plots.
- *PlottingInterPolationRpPb5023GeV.h* & *Interpolation5023GeV.h* are used in the p-Pb interpolation macros and contain specific functions for that as well as the binnings needed for the interpolation.
- *PlottingMeson.h* contains the specific plotting functions for the meson QA needed in the framework.

- **DownloadAndDataPrep**: Contains the shell scripts to download the root-output files created by the lego-trains

per period or per run (Get[GammaConv,GammaCalo,GammaConvCalo,GammaCaloMerged,NeutralMeson]FilesFromGridAndMergeThem_[pp,pPb,PbPb,XeXe][$ENERGY/$PERIOD].sh). The newest versions of these script are for the Run2 p-Pb and Xe-Xe data, which already depend on the (*basicFunction.sh*). In this shell script the commonly used functions for the download are stored for usage in other shell scripts. In the future please use these functions as they already check the consistency of the file and prevent multiple downloads, if not explicitly desired. Furthermore, the directory contains the helper macros for the download to separate files into their different directories (*SeparateDifferentCutnumbers.C*), to change the directory name to the default name (*ChangeStructureToStandard.C*) and to combine the output from different folders into one file (*CombineDifferentFolders.C*). In addition, a number of perl-scripts are provide to create the disired structure for the added signal processing for certain data sets (*MakeLinks*.C*). Last but not least this directory contains the different run lists and Jet-Jet bin number list in **DownloadAndDataPrep/runlists**. Please make sure that if you create a new data set or analyse a new data set that these are contained under the proper name in this folder.

- **RooUnfold**:
  Contains the necessary inputs to include RooUnfold. You need to compile RooUnfold uisng `make` in this folder once you have downloaded the directory in order to be able run the direct photon extraction properly. Afterwards the path to this directory needs to be included in *CompileCorrectGammaV2.C* including your user-name on your device.

- **ExternalInput**, **ExternalInputpPb**, **ExternalInputPbPb**, **LHC11hInputFiles**:
  These directories contain additional root & text files from external sources (like other analyzers in ALICE from PHOS & EMCal on $\pi^0, \eta\&\omega$), other particles like $\pi^{\pm}, K^{\pm}$ … or theory calculations on the various topics. Furthermore they contain the final results for the neutral mesons once they are prelimary or published.

- **SimulationStudies**, **ToyModels**: These folders contain macros to analyse the pure simulation output created by the task *AliAnalysisTaskGammaPureMC.cxx* in **AliPhysics/PWGGA/GammaConv** as well as some macros to produce Toy-MC simulations to study for instance the effect of the the different resultions in the merged cluster analysis and a simple decay simulation for secondary decay studies.

- **DeprecatedMacros**: Here all macros have been moved which are no longer maintained but have been used to create published or preliminary plots. Additionally some older shell scripts to stear the material budget analysis can be found here.

- **SupportingMacros**: This directory contains a number of small macros, which are currently not need in general for the processing, but might be helpful to visualize some specific parts of the data or create specific input files for the grid analysis.

- **SystematicErrorsNew**, **SystematicErrorsOld**: These folder should contain all systematic errors in a text format which have been used for publication or preliminary creation. They will soon be sorted a bit better and reorganized to ease the understanding of these inputs.

- **TaskFlow**: This folder contains all necessary macros to perform the direct photon flow analysis, for further details please ask **Mike Sas**

- **TaskQA**: This is the dedicated folder for the post-processing macro of the analysis-level QA, they are explained in detail in the section **Quality Assurance and Energy Calibration of Calorimeters**. In order to ease the processing also examples for the configuration files are given in **TaskQA/ExampleConfigurations**.

- **TaskV1**: This folder contains all macros and header files needed to perform the neutral meson ($\pi^0/\eta \rightarrow \gamma\gamma, \gamma e^+, e^-$ & merged), heavy meson (ω, ...) and direct photon analyses, which will be explained in the next sections. Addtionally, it contains macros to perform the material budget analysis and characterize the meson analysis further. Also the macros to perform create the energy-position-correction of the calorimeters are stored here. Once more example configurations for the macors which need a text file as input are provide in **TaskV1/ExampleConfigs**.

As you can imagine the software is evolving, as such some macros may exist in several version (V1,V2..), please make sure to always use the latest version if you start developing on top of something, as at some point the older ones will be deleted. In order to prevent incontinuities or larger disruptions these major deletion processes are kept to a minimum though and we try to do the changes in a more gradual fashion.

Analysis Notes and Papers

It is **recommended** to read through some analysis notes before going through this tutorial to gain some insight into the analysis techniques and the physics observables we are looking for.
Below, all papers are listed which are in the context of this tutorial (neutral mesons and photons).

If you click on the respective links, you will be directed to the internal paper pages of ALICE.
In the field *Article Information* on the respective pages, you will find the section *Link to corresponding note*, where all analysis notes are linked which are relevant for the given paper.

It is **highly recommended** to read the analysis notes concerning 2.76 TeV and 8 TeV for neutral mesons and direct photons, as they were the latest notes to be written and therefore describe the latest developed analysis procedures using the latest available version of the software.

List of Publications and documented Analysis Notes

small systems

- Neutral Mesons @ pp 0.9 + 7 TeV
  - PCM
  - PHOS
- Neutral Mesons @ pp 2.76 TeV
  - EMCAL
  - PCM-EMCAL
  - Merged EMCAL
  - Combination
- Neutral Mesons @ pp 8 TeV
  - PCM
  - EMCAL
  - PCM-EMCAL
  - PHOS
  - Combination
- Neutral Mesons @ pPb 5.023 TeV
  - PCM
  - EMCAL
  - PCM-EMCAL
  - PHOS
  - PCM-PHOS
  - Combination
- Direct Photons @ pp 2.76 + 8 TeV
  - Note 2.76 TeV
  - Note 8 TeV

heavy-ions

- Neutral Pions @ PbPb 2.76 TeV
  - pp+PbPb PCM
  - pp PHOS
  - PbPb PHOS

- Direct Photons @ PbPb 2.76 TeV
  - PCM
  - PHOS
  - Significance of Excess
- Direct Photon Flow @ PbPb 2.76 TeV

GRID and AliRoot/AliPhysics

GRID

What is the GRID?

You need to have a valid GRID certificate if you want to work with ALICE data. If you don't know what a GRID certificate is, please ask your supervisor as he/she needs to request a certificate for you.

How-To get/install the GRID certificate

Once you have a valid, correctly installed certificate, you should be able to access: MonALISA Repository for ALICE

Install AliRoot/AliPhysics using aliBuild

Please have a look at the following webpage, it provides everything you need to do / you need to know:

- ALICE Software installation
- aliBuild documentation

Unless you are not looking to analyse jets, you may run aliBuild with the following disable options to reduce the time of the build (in general, you won't need GEANT nor fastjet). If you want to analyse jets, just leave out fastjet in the disable section:

> aliBuild -z -w ../sw -d build AliPhysics --disable GEANT3,GEANT4_VMC,fastjet

Some tweaks (optional)

Create the following files with the given contents:

- compile_alibuild.sh

```
currentFolder=`pwd`
cd $HOME/alice/ali-master
aliBuild -z -w ../sw -d build AliPhysics --disable GEANT3,GEANT4_VMC,fastjet
cd $currentFolder
```

- compile_aliphysics.sh

```
currentFolder=`pwd`
cd $HOME/alice/sw/BUILD/AliPhysics-latest-ali-master/AliPhysics
alienv load AliPhysics/latest-ali-master
make -j5 install
if [ $# -eq 0 ]
  then
    ctest --output-on-failure -j 4 && echo All tests OK
fi
cd $currentFolder
```

- compile_aliroot.sh

```
currentFolder=`pwd`
cd $HOME/alice/sw/BUILD/AliRoot-latest-ali-master/AliRoot
alienv load AliPhysics/latest-ali-master
make -j5 install
if [ $# -eq 0 ]
  then
    ctest -R load_library --output-on-failure -j 4 && echo All tests OK
fi
cd $currentFolder
```

- tbrowser.C

```
{
TBrowser a;
}
```

and add the following lines to your *.bash_aliases* (remember to replace YOURUSERNAME):

```
alias ali='alienv load AliPhysics/latest-ali-master'
alias aliunload='alienv unload AliPhysics/latest-ali-master'
alias alienter='alienv enter AliPhysics/latest-ali-master'
alias alipc='. ~/alice/compile_alibuild.sh'
alias alis='cd ~/alice/ali-master/AliPhysics'
alias alib='cd ~/alice/sw/BUILD/AliPhysics-latest-ali-master/AliPhysics/'
alias alic='. ~/alice/compile_aliphysics.sh'
alias aliroots='cd ~/alice/ali-master/AliRoot'
alias alirootb='cd ~/alice/sw/BUILD/AliRoot-latest-ali-master/AliRoot/'
alias alirootc='. ~/alice/compile_aliroot.sh'
alias token='alien-token-init YOURUSERNAME'
alias tb='root -l ~/alice/tbrowser.C'
```

Then, you just need to type

> ali

in the command line and you are in the ALICE software environment. Using

> alipc

you can re-build the whole software chain and using

> alic

or

> alirootc

you can trigger rebuild of AliPhysics and AliRoot respectively, no matter in which directory you currently are in the shell. Also, you can easily enter a TBrowser by just typing

> tb

.

Running Analysis Tasks

You need to have some ALICE data on your disk if you want to run the *Offline* testing. To download files from a data/MC production of ALICE from the GRID, see Download Files from GRID.

Offline

Test with local software and local files

- process runAnalysisConv.C with useGrid = kFALSE.
- make sure you have downloaded valid test files and provided the correct path.
- make sure that you set up the same tasks and make the same settings as you want to use on the train.
- also, make sure that you have installed the versions of AliRoot and AliPhysics that you want to use on the train.

Run the macro runLocal.C (right-click and save as) with

```
root -x -l -b -q 'runLocal.C("LHC12c", kFALSE, 0, 9)'
```

The additional macros needed are CreateArchivChain and CreateArchivChainAOD (right-click and save as both).

The first argument chooses the data set specified in the *runLocal* macro. The second argument selects between '*offline*' / '*GRID testing*' and the two numbers specify which files should be processed (in the example the files from folder '0' until folder '9').

- if you test AODs, you need to specify the AOD cut number. To know it, look into the aodTree in the AliAODGammaConversion.root
- if you have made major changes to the code, you should test the memory consumption (e.g. with 'top'), the processing time (e.g. with 'time') and the output file size (e.g. with 'ls -sh')

If everything works well, no errors are shown and the full output is generated, you may launch a test on GRID (next chapter).

GRID testing

After local testing was successful, try to test by loading all files from the GRID. For this purpose:

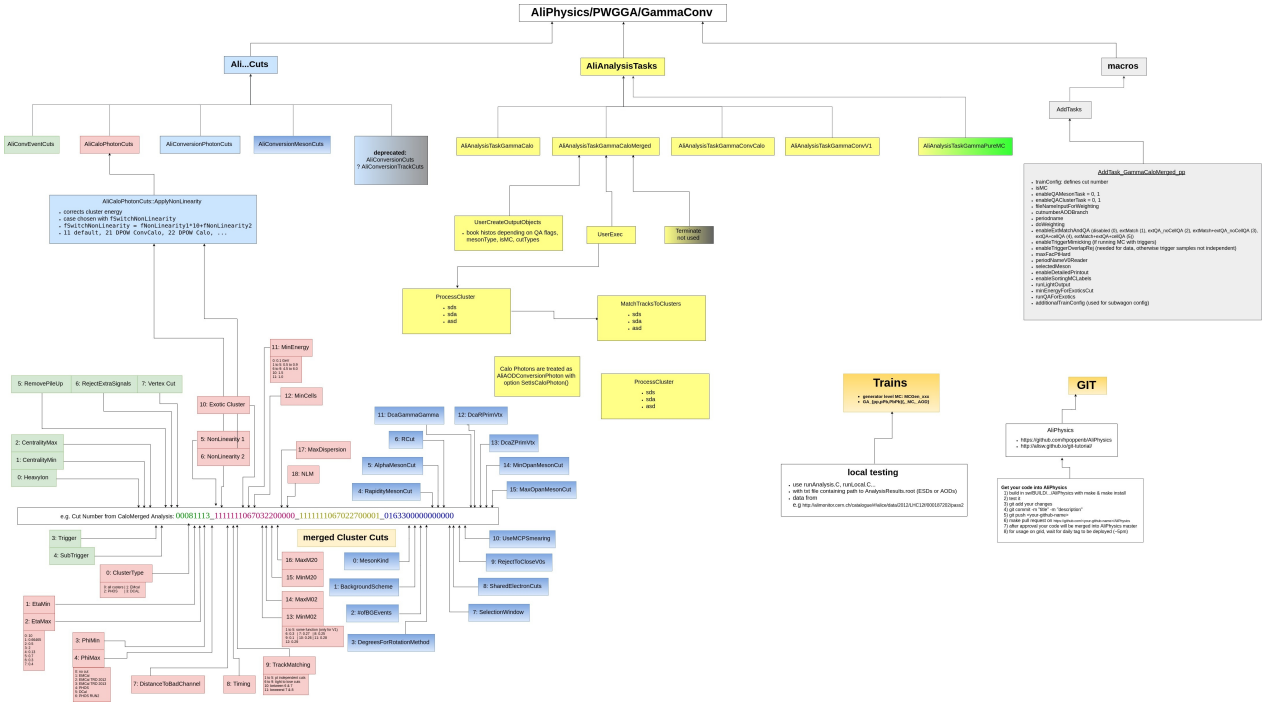
- specify the second argument useGrid == kTRUE for *runLocal.C*
- specify the path to the test files on the GRID in *runLocal.C* (use aliensh to make sure you have specified the correct file path)
- possibly check the settings of the GRID handler in *runLocal.C* (which tag you are using etc)

If something goes wrong, check the stderr for errors and wn.xml if the correct file was used for testing.

- keep in mind that some errors types are only visible during the GRID testing step (like streamer errors etc), therefore this kind of testing is very important!

If everything works and all outputs are fine, you have successfully tested your AnalysisTask locally.

Overview of GammaConv-Software Package & Supporting Classes in AliPhysics



More information on cutstrings

To handle all the different possibilities to select our photons and neutral meson candidates we make use of a "cutstring" for the event cuts, conversion cuts, calo cuts and meson cuts. Changing for example a minimum energy threshold for your analysis will be done by changing a digit in the corresponding cutstring. It is recommended to open the classes in parallel so you can easily navigate around to find which digit corresponds to which cut.

Conversion cut:

```
cuts.AddCut("80000113", "00200009327000008250404000", "0162103500900000");
```

Calo cut:

```
cuts.AddCut("80000113","111114105703223000 0","01631031000000d0");
```

ConvCalo cut:

```
cuts.AddCut("82400113","00200009327000008250400000","1111141057032230000","0163103100000010");
```

Explanation of the cutstrings:

```
//================================================================
AliConvEventCuts.cxx

const char* AliConvEventCuts::fgkCutNames[AliConvEventCuts::kNCuts] = {
  "HeavyIon",                   //0
  "CentralityMin",              //1
  "CentralityMax",              //2
```

```
  "SelectSpecialTrigger",         //3
  "SelectSpecialSubTriggerClass", //4
  "RemovePileUp",                 //5
  "RejectExtraSignals",           //6
  "VertexCut",                    //7
};

//================================================================
AliConversionPhotonCuts.cxx

const char* AliConversionPhotonCuts::fgkCutNames[AliConversionPhotonCuts::kNCuts] = {
  "V0FinderType",           // 0
  "EtaCut",                 // 1
  "MinRCut",                // 2
  "EtaForPhiCut",           // 3
  "MinPhiCut",              // 4
  "MaxPhiCut",              // 5
  "SinglePtCut",            // 6
  "ClsTPCCut",              // 7
  "ededxSigmaCut",          // 8
  "pidedxSigmaCut",         // 9
  "piMomdedxSigmaCut",      // 10
  "piMaxMomdedxSigmaCut",   // 11
  "LowPRejectionSigmaCut",  // 12
  "TOFelectronPID",         // 13
  "ITSelectronPID",         // 14 -- new ITS PID
  "TRDelectronPID",         // 15 -- new TRD PID
  "QtMaxCut",               // 16
  "Chi2GammaCut",           // 17
  "PsiPair",                // 18
  "DoPhotonAsymmetryCut",   // 19
  "CosinePointingAngle",    // 20
  "SharedElectronCuts",     // 21
  "RejectToCloseV0s",       // 22
  "DcaRPrimVtx",            // 23
  "DcaZPrimVtx",            // 24
  "EvetPlane"               // 25
};

//================================================================
AliCaloPhotonCuts.cxx

const char* AliCaloPhotonCuts::fgkCutNames[AliCaloPhotonCuts::kNCuts] = {
  "ClusterType",          //0   0: all,    1: EMCAL,   2: PHOS
  "EtaMin",               //1   0: -10,    1: -0.6687, 2: -0,5, 3: -2
  "EtaMax",               //2   0: 10,     1: 0.66465, 2: 0.5,  3: 2
  "PhiMin",               //3   0: -10000, 1: 1.39626
  "PhiMax",               //4   0: 10000, 1: 3.125
  "NonLinearity1"         //5
  "NonLinearity2"         //6
  "DistanceToBadChannel", //7   0: 0,      1: 5
  "Timing",               //8   0: no cut
  "TrackMatching",        //9   0: 0,      1: 5
  "ExoticCluster",        //10   0: no cut
  "MinEnergy",            //11   0: no cut, 1: 0.05,    2: 0.1,  3: 0.15, 4: 0.2, 5: 0.3, 6: 0.5, 7: 0.75, 8: 1, 9: 1.25
 (all GeV)
  "MinNCells",            //12  0: no cut, 1: 1,       2: 2,    3: 3,    4: 4,   5: 5,   6: 6
  "MinM02",               //13
  "MaxM02",               //14
  "MinM20",               //15
  "MaxM20",               //16
  "MaximumDispersion",    //17
  "NLM"                   //18
};

//================================================================
AliConversionMesonCuts.cxx

const char* AliConversionMesonCuts::fgkCutNames[AliConversionMesonCuts::kNCuts] = {
```

```
  "MesonKind", //0
  "BackgroundScheme", //1
  "NumberOfBGEvents", //2
  "DegreesForRotationMethod", //3
  "RapidityMesonCut", //4
  "RCut", //5
  "AlphaMesonCut", //6
  "SelectionWindow", //7
  "SharedElectronCuts", //8
  "RejectToCloseV0s", //9
  "UseMCPSmearing", //10
  "DcaGammaGamma", //11
  "DcaRPrimVtx", //12
  "DcaZPrimVtx", //13
  "MinOpanMesonCut", //14
  "MaxOpanMesonCut" //15
};

//================================================================
```

How to start with a new data set or Monte Carlo simulation

In the following it will be explained how to get information about the simulation itself and how to obtain the pT hard bin weights that are necessary for your analysis.

Get information about a new data set

There are several ways to get information about a specific dataset. For Run2 data, several new and well maintained sources are available whereas for Run1 data some information is hard to get. In the following, the main sources of information will be presented:

1. On MonALISA select in the sidebar Run Condition Table and select the desired period in the top left (e.g. choose LHC17p) and the reconstruction pass (standard is pass1) on the top right of the table. The run condition table provides already a lot of important information, like the runnumbers, their corresponding filling schemes and fills, the beam energy and the active detector systems in each run. Another important information is the "Physics Selection Status" which signals the general quality of the run from the detector point of view. The run condition table is a quick and easily accessible source of information.
2. The **Data Preparation Group (DPG)** has well maintained information especially for Run2 data. Their main twiki page AliceDPG leads to several subpages with useful information. One of these pages is the DPGDataProcessing page which provides a useful overview of LHC15-LHC17 periods, their collision energies, magnetic field information, interaction rates, reconstruction progress and most important the minimum bias statistics. A similar page, the AliDPGReconstructedDataTakingPeriodsSummary in addition leads to detailed pages for the individual data sets including the **good run number lists**.
3. The **Logbook** provides additional information about each run. Here, detailed information about **trigger configurations** can be found. This, however, requires advanced knowledge of the used abbreaviations and will, for now, not be further explained.

Get information about a new simulation

On MonALISA select in the sidebar Production info/MC production cycles and scroll to the relevant production. For this example, I will use LHC17g8b.

LHC17g8c	Pb-p, 8.16 TeV - JJ events embedded in EPOS-LHC events anchored to LHC16s, CENT+FAST, ALIROOT-7270	Completed	266437-267110	2,908,000	RAW OCDB	18y 53d	34d 22:36	5.101 TB	v5-08-13zc-cookdedx-1
LHC17g8b	p-Pb, 8.16 TeV - JJ events embedded in EPOS-LHC events anchored to LHC16r, CENT+FAST, ALIROOT-7270	Completed	266196-266318	4,084,250	RAW OCDB	25y 157d	40d 13:51	7.101 TB	v5-08-13zc-cookdedx-1
LHC17g8a_fast	p-Pb, 5.02 TeV - JJ events embedded in EPOS-LHC events anchored to LHC16qt, FAST only, ALIROOT-7270	Completed	265309-267166	83,466,000	RAW OCDB	462y 11d	2y 273d	127.5 TB	v5-08-13zc-cookdedx-1
LHC17g8a_cent_woSDD	p-Pb, 5.02 TeV - JJ events embedded in EPOS-LHC events anchored to LHC16qt, CENT, woSDD, ALIROOT-7270	Completed	265309-267166	83,324,500	RAW OCDB	456y 308d	3y 97d	127 TB	v5-08-13zc-cookdedx-1
LHC17g2a	p-p, 13 TeV, Single muon MC anchored to LHC16k, new, ALIROOT-7388	Completed	256504-258537	76,080,000	RAW OCDB	2y 349d	12d 23:41	116.5 GB	v5-09-09-1

On this page you have several important pieces of information already available. You can see if the simulation is "Completed" or still "Running" or currently in "Quality check 10%".

Furthermore, the corresponding run ranges in data to which the simulation is anchored are shown as well as the total statistics produced.

The most important information can be found in the associated Jira ticket which is given by the ending of the second column. In this case "ALIROOT-7270" which is the ending of the link https://alice.its.cern.ch/jira/browse/ALIROOT-7270.

On this Jira page, the full generation information is available. This covers the AliPhysics version, generator information, run lists, intermediate and final QA as well as further information in the comments section.

Description

06.08.2016

- AliPhysics: **v5-08-13zc-01-cookdedx-1**
- AliDPG: **v5-08-XX-28**
- OCDB-snapshots from: **LHC17a2**
- dpgsim: `--generator PWGGA:EPOSLHC_Pythia_GammaTriggerAndJet --process Pythia8Jets`

- **LHC17g8a_cent_woSDD** (anchored to 16qt, CENT)
- **LHC17g8a_fast** (anchored to 16qt, FASTonly)
- **LHC17g8b** (anchored to 16r, CENT+FASTonly)
- **LHC17g8c** (anchored to 16s, CENT+FASTonly)

All sub-cycles are generated without SDD and without ZDC (like the general purpose production **LHC17f2**

For **16r/s**, only runs with TPC complete acceptance and EMCAL ON are generated.

There are 3 runs in **16q** which have the EMCAL OFF but they are simulated. The loss in statistics for PWGGA is ~ 8 Mio / 328 Mio, ie **~2.5%** for these data.

The statistics to be generated is 8 Mio events / pthardbin, integrated over all periods and triggers considered.

From the description you can now see that the simulation is anchored to 16r and 8 Mio events per pT hard bin were simulated as this example is a JetJet Monte Carlo simulation.

In order to get further information about the in this case used generator PWGGA:EPOSLHC_Pythia_GammaTriggerAndJet one has to refer to the Data Preparation Groups Git repository https://github.com/alisw/AliDPG.

In this repository, the used generator config can be found in AliDPG/MC/CustomGenerators/PWGGA/EPOSLHC_Pythia_GammaTriggerAndJet.C where the setting for the process "Pythia8Jets" are set. The process itself is defined in AliDPG/MC/GeneratorConfig.C where we get infromation about the eta, phi, p_T^{hard}, quenchin and structure function that are used for the simulation.

Adding the data set or simulation

In order to add a new data set to the analysis workflow it is always recommended to download a few ESD/AOD files and run local tests with your desired analysis task. This ensures that corrections/calibrations are loaded correctly and the following changes are implemented without compilation errors:

1. For calorimeter analyses, the AliCaloPhotonCuts::FindEnumForMCSet enum can already be defined as it will later on be necessary for the cluster energy calibration (Non-linearity correction).
2. The AliConvEventCuts::SetPeriodEnum() should be defined for the new dataset/MC production.

In the local test it should be checked if calibrations are already available for the data set (e.g. splines, calorimeter calibrations, multiplicity information, ...). If, for example, splines are missing, then running a train on the data set would be suboptimal.

In the first trainrequest for a new dataset ask the train operators to add the dataset or even the META dataset (if several periods are required) to the trainpage. Here, no action besides the request is required.

Extracting p_T^{hard} bin weights

This section becomes important if you need to work with a new JetJet Monte Carlo simulation.

Due to the nature of the JetJet simulation which requires a jet with a transverse energy of at least 5 GeV, several trial are necessary (N_trials). For each event, Pythia also calculates the cross-section to which the gernerated sample of events corresponds on average. As only a small part of the full phase space for each p_T^{hard} bin is sampled, a certain weight has to be applied. The weight is calculated as $\omega = \frac{\sigma_{evt}}{N_{trials}/N_{evt.gen.}}$.

In order to obtain the weights, GammaConv or GammaConvCalo task output is required for each p_T^{hard} bin. These can either be obtained from the Legotrains or from a local test where for each p_T^{hard}bin at least three input files were used.

The files are fed into TaskV1/PlotJetJetMCProperties.C in our AnalysisSoftware repository via an input text that is structured as follows:

```
LHC17g8c/266437/1_GammaCalo_345.root 5 7
LHC17g8c/266437/2_GammaCalo_345.root 7 9
LHC17g8c/266437/3_GammaCalo_345.root 9 12
LHC17g8c/266437/4_GammaCalo_345.root 12 16
LHC17g8c/266437/5_GammaCalo_345.root 16 21
LHC17g8c/266437/6_GammaCalo_345.root 21 28
.....
```

In each line the input file for the given p_T^{hard}bin is given followed by the lower and upper p_T^{hard} limits of this bin. The macro can then be run via:

```
root -x -l -b -q 'TaskV1/PlotJetJetMCProperties.C+("pPbJJWeightsLHC17g8c.txt","80000513_2444400051013200000_016310310000
0010",5,20,"pdf","pPb_8TeV","LHC17g8b",kFALSE)'
```

where the input text file must be given, the cutnumber in the GammaCalo or GammaConvCalo output, the mode (in this case PHOS), the total number of p_T^{hard} bins (here 20), the energy (here "pPb_8TeV"), the MC period (here "LHC17g8b") and a QA flag.

Running the macro will report the individual p_T^{hard} bin weights in the terminal, see:

```
>>>>>>>> pT hard bin #: 0
ntrials: 1000     xSection: 26.3885     number of generated events: 1000     weight: 26.3885     weight applied: 1
80010113_1111100017032230000_01631031000000d0    80010113        1111100017032230000        01631031000000d0
-> found TopDir: GammaCalo_201
using rapidity of 0.8
>>>>>>>> pT hard bin #: 1
ntrials: 1000     xSection: 8.16088     number of generated events: 1000     weight: 8.16088     weight applied: 1
80010113_1111100017032230000_01631031000000d0    80010113        1111100017032230000        01631031000000d0
-> found TopDir: GammaCalo_201
using rapidity of 0.8
>>>>>>>> pT hard bin #: 2
ntrials: 500     xSection: 3.93751     number of generated events: 500     weight: 3.93751     weight applied: 1
80010113_1111100017032230000_01631031000000d0    80010113        1111100017032230000        01631031000000d0
-> found TopDir: GammaCalo_201
...
```

This results, in the example case, in a weight of 26.3885 for the first p_T^{hard} bin, 8.16088 for the second bin, and so on. The weights then need to be added to AliPhysics/PWGGA/GammaConv/AliConvEventCuts.cxx. For this, a period enum needs to be set for the JJ MC simulation in AliConvEventCuts::SetPeriodEnum(). The enums are defined in the header and, of course, the new enum must be added there in addition. The enum then also needs to be set in AliConvEventCuts::GetXSectionAndNTrials() and AliConvEventCuts::GetPtHard().

Then, in AliConvEventCuts::IsJetJetMCEventAccepted() the weights can be set for the respective period similar to:

```
else if ( fPeriodEnum == kLHC17g8c ){
    Double_t ptHardBinRanges[21]  = { 5, 7, 9, 12, 16, 21, 28, 36, 45, 57, 70, 85, 99, 115, 132, 150, 169, 190, 212, 235
, 10000};
    Double_t weightsBins[20]      = {2.638850E+01, 8.160880E+00, 3.937510E+00, 1.485000E+00, 5.382460E-01, 2.034610E-01,
 6.293600E-02, 2.206170E-02, 9.319700E-03, 3.354230E-03, 1.392300E-03, 5.023470E-04, 2.645860E-04, 1.299660E-04, 6.41531
0E-05, 3.469890E-05, 1.816550E-05, 1.047480E-05, 5.728760E-06, 8.547820E-06};
    Int_t bin = 0;
    while (!((ptHard< ptHardBinRanges[bin+1] && ptHard > ptHardBinRanges[bin]) || (ptHard == ptHardBinRanges[bin]) ) )bi
n++;
    if (bin < 20) weight = weightsBins[bin];
}
```

The bin ranges (listed in the JIRA respective ticket) and their weights must be added with the last bin going to "infinity" (e.g. 10,000). When these changes are comitted to AliPhysics and available in a new Tag, trains can be run as usual and the weights will be applied.

AliConvEventCuts::GetUseNewMultiplicityFramework() for all run2 heavy ion datasets/MC

LEGO trains

The LEGO framework is a tool to run and manage analysis trains on AliEn. It builds on existing infrastructure, the analysis framework, MonALISA and LPM. LEGO provides a web interface for users and operators. All details and an extensive How-To may be found here: Analysis Trains with the LEGO Framework

An overview of all LEGO trains may be found here: LEGO trains

Configure a LEGO train

Always test your setup locally before submitting/configuring an analysis train (Running AnalysisTasks). This will make it much easier for you to spot problems/errors.

In Running AnalysisTasks, a working example and a complete setup of main (GammaCalo, GammaConvV1,...) and basic tasks (CDBconnect, PhysicsSelection, PIDResponse etc) is given which is needed to run the analysis using AliPhysics.

Make sure that you set the correct dependencies in the wagons you are creating. Always have a look at existing wagons and use the cloning feature if new wagons need to be add. Make sure to use the subwagon feature. Double-check the set of parameters in the wagons which are handed to the AddTasks.

Request a LEGO train

- make sure that you have tested that your code and wagon configuration works on the dataset that you want to run a train on (see Running AnalysisTasks)
- make sure that all your requested wagons are already activated for the desired datasets!
- send a mail to the mailing list *alice-pwg-GA-train-operators* with the following information:
    - the train (e.g. GA_PbPb_AOD)
    - the wagons; including the wagons that your analysis wagons have dependencies on. Check if your wagons have the correct output file name specified. If neccessary, ask the train operators to add the desired name to the list. Check if all wagons have the right arguments (the ones that you have tested locally), separated by commas. Check that there are no extra commas at the beginning or the end. Check if all wagons have the right dependencies specified. Activate your wagons for the dataset you want to run your train on
    - dataset; check if the dataset is already in the list of datasets on the train page, if not ask to have it added. If you are running over a dataset for the first time, check also if the dataset is defined correctly (Reference production, global variables if used in your wagons, friend chain name "AliAODGammaConversion.root" for AODs, "AOD production"="AOD with ESD" or "AOD production", TTL, SplitMaxInputFileNumber, set Number of Files to Test depending on how many events are saved in one file so that you have a few hundred test events)
    - runlist; check that it is comma separated and that there are no extra commas at the beginning or the end.
    - if you want to have the output runwise. You might need this for QA.
    - aliphysics version (e.g. vAN-20161118-1)
    - if you want to use the slow train option
    - a comment that describes the train, e.g. the purpose or what you changed with respect to the last train on this dataset

an example, title of email: *Train request GA_PbPb*

```
train:                  GA_PbPb
wagons:                 PhotonQA\_5TeV\_0090, PhysicsSelectionESD, CentralityTaskESD2015, PIDResponseTaskESD2015
dataset:                LHC15o\_pass1\_highIR
runwise:                yes
runlist:                CentralBarrelGood
aliphysics:             vAN-20171024
slow-train-option:      yes
comment:                PhotonQA
```

- in case train test is successful (a green 'OK' in the test results)
  - download output (via 'merge dir') and check if all relevant histograms for your analysis are filled and look reasonable
  - check if train operator has made all settings correctly according to your request
  - check memory consumption and processing time per event in the test results (there should be no red numbers! Memory should be below 2GB. If the delta values are problematic it could be due to too few test events)
  - check the testing output log if necessary (e.g. if the correct splines are loaded)
- in case of errors in the train test
  - check stdout, stderr and the testing output log
  - if necessary, redo local and grid tests on the files that were used for the train (see testing output log) and try to reproduce and debug the error (set the debug mode of the AliAnalysisManager to kDebug)
  - if you need help, ask the train operators, the alice-project-analysis-task-force, alice-analysis-operations or one of the train experts
- when the train is finished
  - check the success of the processing jobs (via 'processing process') and the merging jobs (via 'merging process'). In case of failed jobs, click on the PID (=process ID) to have a look at the trace and the log files. This way, check if crashes are due to grid errors or due to bugs in the code which have to be debugged.
  - check the histogram 'Input files per job'. It is not efficient when most jobs have only one or two input files. If this is the case, ask the train experts to redistribute the files
  - report on broken MC files
  - download the output, see Download Files from GRID

Download Files from GRID

There are two (different) approaches to download files from the GRID. Using a C++ macro (1.) and using a bash script (2.). Both approaches have their (dis-)advantages, you may decide what is more convenient for you:

1. via C++ macro

Download ESD/AODs from the GRID (data / MC production)

The download macro may be found here: DL-macro (right-click and save as). It can be called with:

```
root -x -l -b -q 'GridJobFileListDL.C("list1.txt","pp/LHC12c/pass2","YOURPATH/LocalFiles")'
```

It automatically downloads the specified files, saves them in the folder structure needed for running local tests (-> Running AnalysisTasks) and unzips the compressed files.

The first argument is the list of files you want to download, an example content of list1.txt:

```
/alice/data/2012/LHC12c/000180720/pass2/12000180720015.126/root_archive.zip
/alice/data/2012/LHC12c/000180720/pass2/12000180720046.105/root_archive.zip
/alice/data/2012/LHC12c/000180717/pass2/12000180717055.11/root_archive.zip
/alice/data/2012/LHC12c/000179920/pass2/12000179920020.51/root_archive.zip
/alice/data/2012/LHC12c/000182322/pass2/12000182322020.13/root_archive.zip
/alice/data/2012/LHC12c/000179916/pass2/12000179916001.12/root_archive.zip
/alice/data/2012/LHC12c/000182687/pass2/12000182687019.37/root_archive.zip
/alice/data/2012/LHC12c/000182687/pass2/12000182687013.36/root_archive.zip
/alice/data/2012/LHC12c/000182299/pass2/12000182299050.31/root_archive.zip
/alice/data/2012/LHC12c/000182730/pass2/12000182730037.22/root_archive.zip
```

The second argument specifies the relative path where data should be stored (-> need to be specified in *runLocal.C* Running AnalysisTasks) and the third argument gives the absolute path on your system.

There is a possibility to hand over a fourth argument to the function, if you want to download the files from a specific SE (in case the download is slow due to some unresponsive SEs, for exaple '*ALICE::FZK::SE*').

Download LEGO train output (analysis)

All the macros to download LEGO train output files can be found in *AnalysisSoftware/TaskQA*.

Examples how to use can be found within the macros. The macros are able to deal with normal LEGO train output as well as output from META-datasets trains (both examples are contained in the macros) - in the latter case you MUST download the output from each child separately, the full merging does not do what you thin.

- Grid_CopyFiles.C

```
root -x -b -l -q 'TaskQA/Grid_CopyFiles.C+("pp", "ESD", "YOURPATH")'
```

The three arguments need to be adjusted to what you are downloading and where to save the files -> YOURPATH. Inside the macro, specifiy how many data trains and how many different trains you are downloading, the data set names, the output directory, the train run numbers, the desired runlists, the files to download and which files from which sets to merge.

- Grid_CopyFilesJetJet.C

The core of this macro is identical to the first one, but with additional features to download JetJet output per run and per momentum bin. All settings can be deduced from the macro itself.

- Grid_CopyFiles_Runwise.C

Again, the same as *Grid_CopyFiles.C* but performs the runwise download of data from LEGO trains on the GRID. It reads the runlists from *DownloadAndDataPrep/runlists* by the data set names you specify.

- Grid_FailedMergeDLRun.C

If some merging jobs failed on GRID, you can use this macro to download the files and run the merging locally. Then you can merge with the output file from the LEGO train in order to get the full statistics.

- Grid_CheckFiles_Missing.C

This macro checks the input for "missing tracks" in the context of AOD processing of PCM photon candidates with deltaAODs.

- Grid_ReadJetJet.C

This macro can read in bins/runs of a JetJet MC and plots the NTrials and XSection.

2. via bash script

Download LEGO train output (analysis)

Over the past few years we also implemented quite some shell scripts to download the data from the grid in particular for the lego train outputs, they can be found in AnalysisSoftware/DownloadAndDataPrep. They are usually data set & analysis method specific and take care of the correct download and merging of the corresponding lego train outputs. As such they are in a lot of cases very specific and might need a bit more adjustments when adapting for a new data set. At the same time they also allow very specific modifications (i.e. adjusting the download and merging of specific run-lists only). Furthermore, also the download of partially unmerged runs is slightly easier and the number of downloads can be a bit optimized by for instance always downloading the full *root_archive.zip* and only deleting the corresponding files in the end. The download scripts are named according to the following scheme:

- GetGamma[Calo,Conv,ConvCalo,CaloMerged]FilesFromGridAndMergeThem_[$COLLSYS/$ENERGY/$PERIODNAME].sh i.e `GetGammaCaloFilesFromGridAndMergeThem_PbPbLHC10h.sh` or `GetGammaConvCaloFilesFromGridAndMergeThem_pPbRun2_5TeV.sh`

As the script scripts got better we recommend to always have a look at the lastest scripts which have been added to the repository, as these most likely contain the most recent features. Currently the best developed are the ones for `pPbRun2_5TeV` & `XeXe`. These are based already on the common functions file *basicFunction.sh* in which all functions, which had been used more frequently have now been implemented. The shell scripts rely on the actual shell interface of alien meaning the programs:

```
alien_ls
alien_cp
alien_mkdir
alien_find
...
```

These programs can be executed directly from the bash once you initialized your alice-environment. As such the parsing of files is sometimes more convenient and consequently more straight forwards to parse. Often also basic shell programs as `sed, cat, grep, find` and so on are used. Thus if you are not sure what a certain option does just type `man $PROGRAMNAME` and have a look at the corresponding man-page directly or ask us.

The most complex call for the scripts is :

```
bash GetGammaCaloFilesFromGridAndMergeThem_pPbRun2.sh fbock 0 fast _FAST _fast
```

However usually only the first two arguments after the script-name are defined. Where the first one defines your username and in accordance with that some global settings, which you need to add to:

```
if [ $1 = "fbock" ]; then
    BASEDIR=/mnt/additionalStorageExternal/OutputLegoTrains/pPb
elif [ $1 = "dmuhlhei" ]; then
    BASEDIR=~/data/work/Grid
fi
```

in the beginning of the script. The corresponding outputs are gonna be stored in this base directory with the folder-name $TRAINDIR. In addition depending on this-base directory ($BASEDIR) the number of slashes to reach a certain file is calculated as follows:

```
# Definitition of number of slashes in your path to different depths
NSlashesBASE=`tr -dc '/' <<<"$BASEDIR" | wc -c`
NSlashes=`expr $NSlashesBASE + 4`
NSlashes2=`expr $NSlashes - 1`
NSlashes3=`expr $NSlashes + 1`
NSlashes4=`expr $NSlashes + 2`
echo "$NSlashesBASE $NSlashes $NSlashes2 $NSlashes3 $NSlashes4"
```

In addition various global variables are set, which allow you to switch on the downloading of the periods, single runs as well as enabling the merging

```
DOWNLOADON=1            # 0/1 - dowload off/on
MERGEON=1               # 0/1 - global merging of multiple periods switched off/on
SINGLERUN=1             # 0/1 - download single runs off/on
SEPARATEON=1            # 0/1 - switch off/on separation of cutnumber into different files
MERGEONSINGLEData=1     # 0/1 - disable/enable merging for single runs of data
MERGEONSINGLEMC=1       # 0/1 - disable/enable merging for single runs of MC
CLEANUP=1               # 0/1 - disable/enable cleaning up of directory
CLEANUPMAYOR=$2         # 0/1 - disable/enable mayor cleaning up of directory handed as second argument to shell-script
number=""               # temporary variable
```

Afterwards the general data set variable are created and the corresponding booleans are enabled. The latter should be 1 if the data set is there and will be automatically put to 0, if they are not.

```
# check if train configuration has actually been given
HAVELHC16q=1
HAVELHC16t=1
HAVETOBUILDData=0
HAVELHC17f2b=1
HAVELHC17f2afix=1

# default trainconfigurations
LHC16qData="";
LHC16tData="";
LHC16qtData="";
LHC17f2bMC="";
LHC17f2a_fixMC="";

passNr="1";
```

Last but not least of the configurations the train-numbers which you want to download are handed as data set names, and the script will parse the corresponding train runs.

```
TRAINDIR=Legotrain-vAN20171005-dirGammaRun2
# woSDD (CENT)
 LHC16qtData="679"; #pass 2
 LHC16qData="child_1"; #pass 3
 LHC16tData="child_2"; #pass 2
 LHC17f2bMC="1090";
 LHC17f2a_fixMC="1088";
```

Then the script checks if the train runs exist and downloads the files as specified. If needed this is done run-wise.

```
 CopyFileIfNonExisitent $OUTPUTDIR_LHC17f2a_fix "/alice/cern.ch/user/a/alitrain/PWGGA/GA_pPb_MC/$LHC17f2a_fixMC/merge" $
NSlashes "" kTRUE
```

For the new pPb data-set (16q,r,s,t) multiple reconstructions are available. Thus, in order to correctly download the run-wise output for MC, the arguments `fast _FAST _fast` , `woSDD _CENT_woSDD _cent_woSDD` or `wSDD _CENT_wSDD _cent` had to be added to the shell script call. The first one defining the runlist-name addition, the second defining the data additional name and the third the MC additional name. Afterwards the files are merged according to certain runlists defined in AnalysisSoftware/DownloadAndDataPrep/runlists

```
 MergeAccordingToSpecificRunlist fileLHC17f2a_fix.txt $OUTPUTDIR_LHC17f2a_fix $NSlashes3 GammaCalo All runlists/runNumbe
rsLHC17f2a_fix_$3_all.txt
```

and brought into a common naming scheme and format, also containing the runlist name.

```
 ChangeStructureIfNeededCalo $fileName $OUTPUTDIR_LHC16q $NSlashes "LHC16q_$3-pass$passNr-All" "-All"
```

Lastly the sub-periods which can be merged are merged using `hadd` as well as the corresponding MC's if need be.

If the second option of the scripts was a `1` the script will only do the cleanup of the runwise outputs, so make sure you really wanna do that, if you set it.

The EMCal correction framework

The full set of EMCal calibrations was previously handled by the Tender and was composed of several tasks that were, by demand, added to the analysis chain. The EMCal group has introduced a new framework for the handling of the corrections/calibrations which in addition brings many more useful features. The framework was thoroughly tested by the group and verified to yield the same results as using the Tender.

Setting up the framework

Setting up the correction framework invovles writing a yaml configuration file. The basic configuration can be found in AliPhysics/PWG/EMCAL/config/AliEmcalCorrectionConfiguration.yaml. In this configuration, the default setting for all corrections/calibrations is "false" meaning that they are not being applied. A very detailed introduction to the parts of this configuration can be found on the AliDoc webpage.

In the following, only the basic adjustments to the configurations will be explained. The current configurations that are available for our tasks can be found in AliPhysics/PWGGA/GammaConv/config/PWGGA_CF_config.yaml and AliPhysics/PWGGA/GammaConv/config/PWGGA_CF_config_MC.yaml.

The basic configuration consists of the definition of the input/output objects for cells, clusters and tracks. In addition, the reconstruction pass is set here, or can be put to an empty string ("") to let the framework auto-detect the pass.

```
configurationName: "PWGGA configuration"
pass: "pass4"
inputObjects:
    cells:
        defaultCells:
            branchName: "usedefault"
    clusterContainers:
        defaultClusterContainer:
            branchName: "usedefault"
        ClusterContainerS500A100W3:
            branchName: "S500A100W3ClustersBranch"
    trackContainers:
        defaultTrackContainer:
            branchName: "usedefault"
```

With the configuration shown above, the default cells, cluster and track objects are defined. These are the object which also the previously utilized Tender used. In addition, it is possible to define additional branches/containers which can have different settings. These must have custom names, (i.e. ClusterContainerS500A100W3 with the branchName S500A100W3ClustersBranch) and can be made for cells, clusters and tracks. It is possible to define as many of these objects as desired.

As for us currently only different clusterizers are of interest, we do not use custom cell or track objects. For data, we apply the energy, bad channel and timing corrections. As shown in the example below, there is the default configuration "CellEnergy" as well as "CellEnergy_defaultSetting". This setting is necessary to not apply the corrections more than once. Every setting in "CellEnergy" would be applied in addition in all other custom configuration (more on that later).

```
CellEnergy:
    createHistos: true
CellEnergy_defaultSetting:
    enabled: true

CellBadChannel:
    createHistos: true
CellBadChannel_defaultSetting:
    enabled: true

CellTimeCalib:
    createHistos: true
CellTimeCalib_defaultSetting:
    enabled: true
```

Therefore, the corrections are only enables in the "defaultSetting" configuration which will always be run before any other custom setting.

After the cells, the clusterizer settings are made. As we have defined an additional output object above, we also need to define an additional clusterizer for this object, see below:

```
Clusterizer:
    createHistos: true
    clusterizer: kClusterizerv2
    cellTimeMin: -500e-9
    cellTimeMax: +500e-9
    clusterTimeLength: 1e6
    recalDistToBadChannels: true
    remapMcAod: false
    diffEAggregation: 0.0
    cellsNames:
        - defaultCells
Clusterizer_defaultSetting:
    # w0 standard is 4.5
    enabled: true
    cellE: 0.1
    seedE: 0.5
    clusterContainersNames:
        - defaultClusterContainer
Clusterizer_S500A100W3:
    enabled: true
    cellE: 0.1
    seedE: 0.5
    w0: 3
    clusterContainersNames:
        - ClusterContainerS500A100W3
```

As seen in the setting above, we again have a "Clusterizer" and "Clusterizer_defaultSetting" configuration. Every variable set in "Clusterizer" will be also set for all other clusterizers. Meaning the setting of the v2 clusterizer, the +-500ns timing cuts, etc. The "defaultSetting" clusterizer then makes clusters with our -usually- standard settings of 500MeV seed, 100MeV aggregation and a w0 parameter of 4.5. The setting for our additional output object "S500A100W3" is different just by the setting of `w0=3` . For the clusterizer, many different settings are available and can be looked up in the default configuration AliPhysics/PWG/EMCAL/config/AliEmcalCorrectionConfiguration.yaml.

After the clusterizer, the nonlinearity correction can be applied. As we apply our own correction in AliCaloPhotonCuts.cxx, the setting here is kNoCorrection and must be applied for all configurations.

```
ClusterNonLinearity:
    createHistos: true
    nonLinFunct: kNoCorrection
ClusterNonLinearity_defaultSetting:
    enabled: true
    clusterContainersNames:
        - defaultClusterContainer
ClusterNonLinearity_S500A100W3:
    enabled: true
    clusterContainersNames:
        - ClusterContainerS500A100W3
```

Local running

For a local test, using the correction framework we now have to add the corresponding tasks for our default setting and the final setting that we want to use in the analysis. This would look like the following in the local test macro:

```
const UInt_t  kPhysSel  = AliVEvent::kAnyINT | AliVEvent::kCentral | AliVEvent::kSemiCentral;
AliEmcalCorrectionTask::BeamType iBeamType = AliEmcalCorrectionTask::kpp;
AliEmcalCorrectionTask * correctionTask = AliEmcalCorrectionTask::AddTaskEmcalCorrectionTask("defaultSetting");
   correctionTask->SelectCollisionCandidates(kPhysSel);
   correctionTask->SetForceBeamType(iBeamType);
   correctionTask->SetUserConfigurationFilename("PWGGA_CF_config.yaml");
   correctionTask->Initialize();
AliEmcalCorrectionTask * correctionTaskSpezial = AliEmcalCorrectionTask::AddTaskEmcalCorrectionTask("S500A100W3");
   correctionTaskSpezial->SelectCollisionCandidates(kPhysSel);
   correctionTaskSpezial->SetForceBeamType(iBeamType);
   correctionTaskSpezial->SetUserConfigurationFilename("PWGGA_CF_config.yaml");
   correctionTaskSpezial->Initialize();
```

The task needs to be defined with the configuration name handed as argument. Basic settings must be made and the configuration file needs to be set. Each task then needs to be initialized to properly apply the settings.

The above configuration would add the following configurations and components. Notice, how the cell corrections are only applied once and in "defaultSetting".

```
I-AliEmcalCorrectionTask::InitializeComponents: Successfully added correction task: AliEmcalCorrectionClusterNonLinearit
y_defaultSetting
AliEmcalCorrectionTask_defaultSetting Settings:
Correction components:
    AliEmcalCorrectionCellEnergy_defaultSetting
    AliEmcalCorrectionCellBadChannel_defaultSetting
    AliEmcalCorrectionCellTimeCalib_defaultSetting
    AliEmcalCorrectionClusterizer_defaultSetting
    AliEmcalCorrectionClusterNonLinearity_defaultSetting

I-AliEmcalCorrectionTask::InitializeComponents: Successfully added correction task: AliEmcalCorrectionClusterNonLinearit
y_S500A100W3
AliEmcalCorrectionTask_S500A100 Settings:
Correction components:
    AliEmcalCorrectionClusterizer_S500A100W3
    AliEmcalCorrectionClusterNonLinearity_S500A100W3
```

This now allows us to use the new configuration "S500A100W3" in our analysis tasks via: `AliAnalysisTask *taskEMC = AddTask_GammaCalo_pp(0,intMCrunning,1,1, "",conversionAODCutnumber ,periodName,kFALSE,kFALSE,5,kFALSE,kTRUE,3.,periodName,kFALSE,"","",kTRUE,kFALSE,"S500A100W3", "450" );`

Grid running

The correction framework can also be used on the grid, however, this part is still under development and will be added in the future.

Overview

This chapter is about the very first and highly important step of performing any analysis: the **Quality Assurance (QA)** of recorded data and Monte Carlo (MC) productions.

It is important to ensure that all detectors behave as expected and that related observables (a couple of examples: Number of Tracks in TPC, dE/dx of electrons, Number of calorimeter clusters found,...) are described properly by MC simulations. In particular, it is important to check that any of these observables is the same for all the subsets of the data, namely for all different runs that were recorded. If this is not the case, the differences *must* be understood in case the MC does not follow.

The QA framework can be found within the AnalysisSoftware Repository in *TaskQA/** , it is split into the major parts:

1. Event QA
2. Photon QA
3. Cluster QA
4. PrimaryTrack QA

As data taking is split into runs, a so-called runwise QA must be run on the desired data/MC to analyze. Furthermore, it is important to globally check the observables for the full statistics.

> The respective macros can be found in *TaskQA/EventQA.C* and *TaskQA/EventQA_Runwise.C* (and accordingly for the other three examples)

Generally, all calorimeter related macros/functions should be able to handle EMCal as well as PHOS and DCal, but for last two calorimeters special care needs to be taken and every QA step still needs to be verified in detail (after sucessful validation this statement may be removed).

GRID running

All QA tasks need to have the *slow* option active as well as *runwise processing* must be enabled on GRID (remember to include this in your request Requesting a LEGO Train).

- The PhotonQA is run by specialized wagons running the AddTask_PhotonQA, they can be found on the MonAlisa pages in the *Group PCM* - look for *PhotonQA**.
- The EventQA needs the standard output by one of our standard analysis tasks (GammaConvV1, GammaCalo, GammaConvCalo,..)
- For ClusterQA peferably run the standard calorimeter-related tasks GammaCalo & GammaConvCalo (as you will need output from both to verify the energy calibration) - if bad cell identification should be performed in addition, one of both tasks need to be run with QA level, for example '5' to enable full QA output including cell QA

Two different sets of download macros are available in *DownloadAndDataPrep/** and *TaskQA/Grid_CopyFiles**, see Download Files from GRID to download the runwise output as well as the fully merged output from GRID.

Processing

Always run the runwise macros first, then the merged output (also referred to as periodwise or full output)!

For runwise processing with the QA framework, the *Gamma*.root* files should be placed in the following folder scheme in the AnalysisSoftware folder (AnalysisSoftware Repository):

> DataQA/DATE/PERIODNAME/RUNNUMBER/Gamma*.root (for example: DataQA/20180104/LHC15h1b/177681/GammaCalo_104.root and so on for the differnt runs available for the given dataset)

Furthermore, all screen output is saved in **.log* files - watch for them if you want to look up any information.

Make sure that a file *runlistsPERIODNAME.txt* (for example runlistsLHC15h1b.txt) is contained in *DownloadAndDataPrep/runlists* containing the different run numbers line by line (examples may be found when checking out from AnalysisSoftware Repository) as the QA framework will look for the run numbers in that folder per default.

Central Steering Macros

The central steering macros for the QA framework are:

- QAV2.C
- QA_RunwiseV2.C

which can be called from bash via:

```
root -x -l -b -q TaskQA/QAV2.C+(\"config.txt\"\,kTRUE\,kTRUE\,kTRUE\,kTRUE\,kTRUE\,2\,\"eps\")
```

and

```
root -x -l -b -q TaskQA/QA_RunwiseV2.C+(\"config.txt\"\,kTRUE\,kTRUE\,kTRUE\,kTRUE\,kTRUE\,2\,\"eps\")
```

Example configurations for *config.txt* may be found within *TaskQA/ExampleConfigurations* while the other parameters of the functions stand for:

- doEventQA -> *kTRUE*
- doPhotonQA -> *kTRUE* (set *kFALSE* if you are not using any photon conversions)
- doClusterQA -> *kTRUE* (set *kFALSE* if you are not using any calorimeter clusters)
- doMergedQA -> *kTRUE* (set *kFALSE* if you are not running a merged analysis)
- doPrimaryTrackQA -> *kTRUE* (set *kFALSE* if you are not running Dalitz or neutral meson analysis via 3 pions)
- doExtQA -> *2* (*0* -> switched off, *1* -> normal QA mode, *2* extendedQA mode with cell QA)
- suffix -> *eps* (*pdf*,*C*,...)

IMPORTANT FOR RUNNING PHOTON QA

As we are using TTrees for the photon QA, there is a slightly different approach in this case: ***only* download runwise output for the photon QA, as in most cases the full merges *will* fail on GRID due to huge output size!**

> during processing of PhotonQA_Runwise.C, the periodwise output will be automatically produced!

Energy Calibration of Calorimeters

Furthermore, the energy calibration of calorimeters directly follows the QA stage and is described in

- Energy Calibration of Calorimeters

> the macros are contained in AnalysisSoftware Repository in TaskV1/*

In general, the calibration of calorimeters is performed by the specific detector groups for EMCal/DCal and PHOS. What we are referring to in this chapter is an improved energy calibration scheme based on the measured neutral pion peak position in data to which the simulated MC mass positions are tuned to. In general, it may happen that an improved energy calibration is not needed but still then the macros should be run to cross-check the calibration.

EventQA

This part of the QA must be run for any analysis as it checks basic event properties.

The event QA covers general event properties as well as global variables like:

generated histograms (list of examples) (full set of generated histograms can be deduced from the macros themselves/or from the output generated):

1. Cut overview histograms (vs. p_T)
2. Number of events, number of tracks in TPC,...
3. Fraction of MinBias events, fraction of good events, fraction of pileup events...
4. Z-vertex distributions and its mean+sigma values for each run
5. Number of Photon Candidates per event: PCM, Calo
6. Neutral meson related quantities: avg. number of meson candidates per event, mean/sigma of transverse momentum, Y, alpha, opening angle of decay gammas(in case of 2 gamma analysis)...

Running the EventQA(_Runwise).C will save the output into the following folder structure:

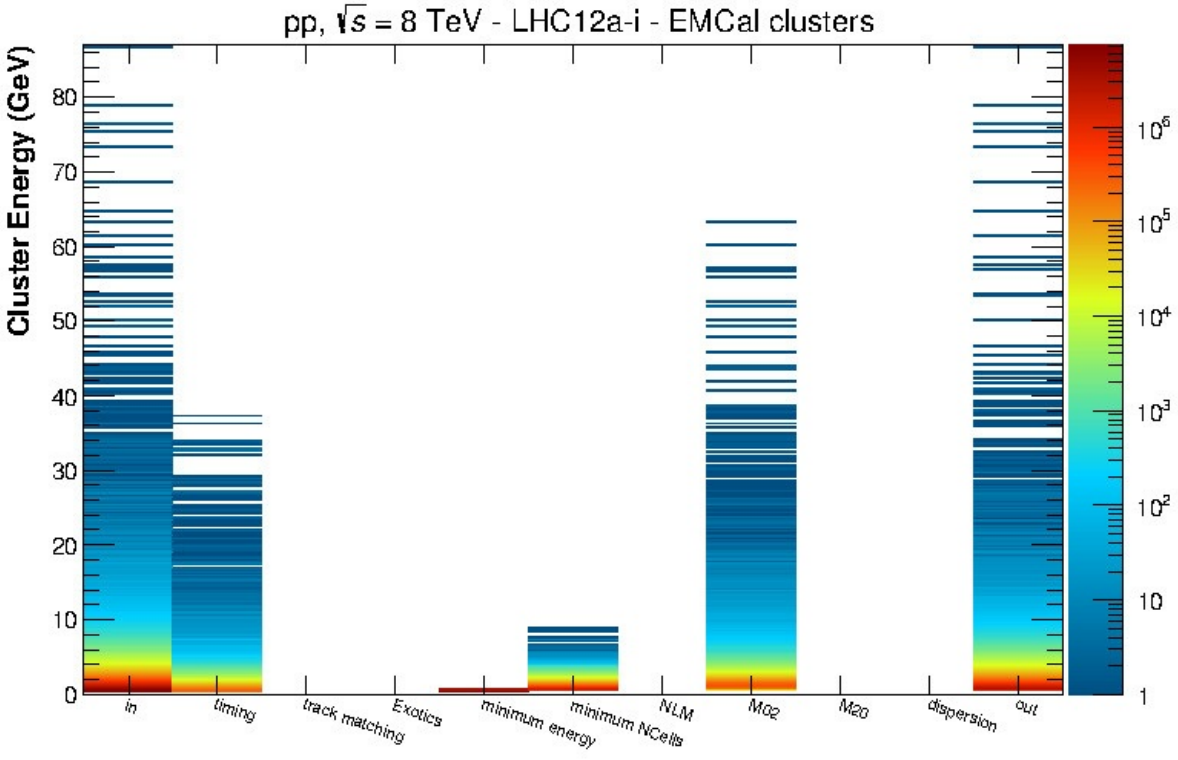
> *CUTNUMBER/SYSTEM/EventQA/*

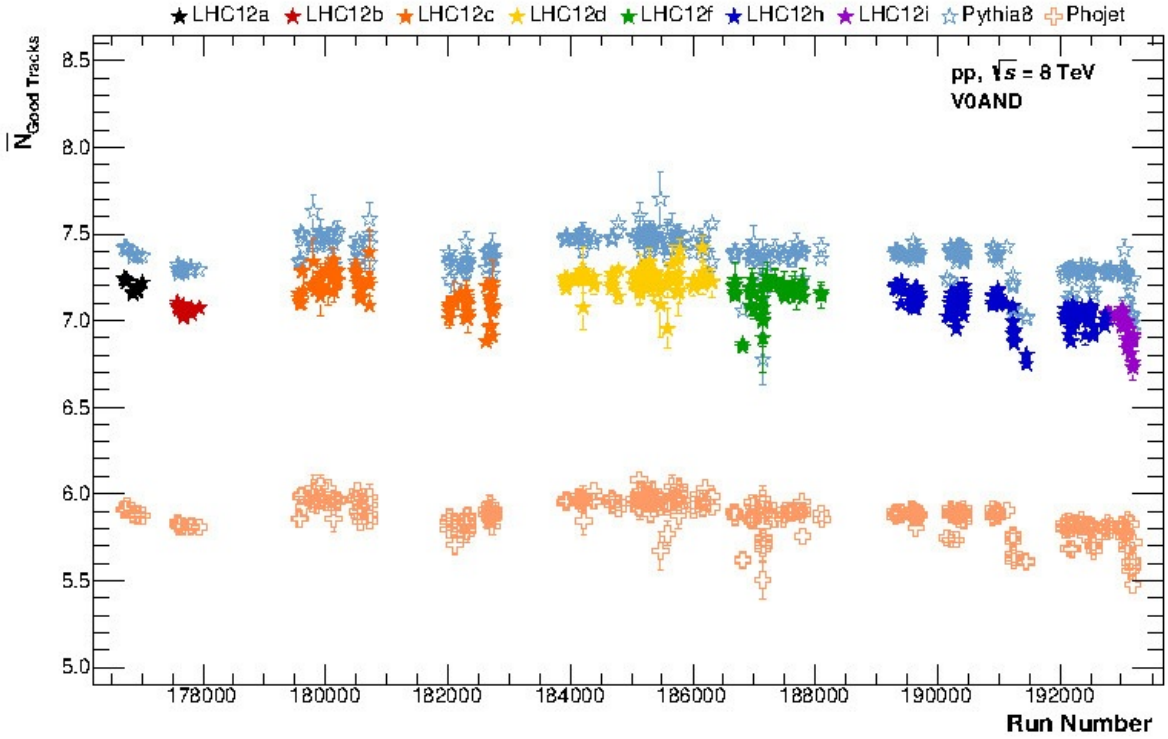
In addition, *.root files will be generated in *CUTNUMBER/SYSTEM/* containing all the histograms as well.

Carefully check all output from runwise/full output with special focus on data/MC comparison (Is the MC able to reproduce all QA histograms extracted from data? Does the MC follow the trends seen in data? Are there any suspicious runs or any observations that cannot be explained?...) In general, they should be stable vs. run number - however, one of the exceptions is pileup which may vary from run to run -> need to be taken with special care!

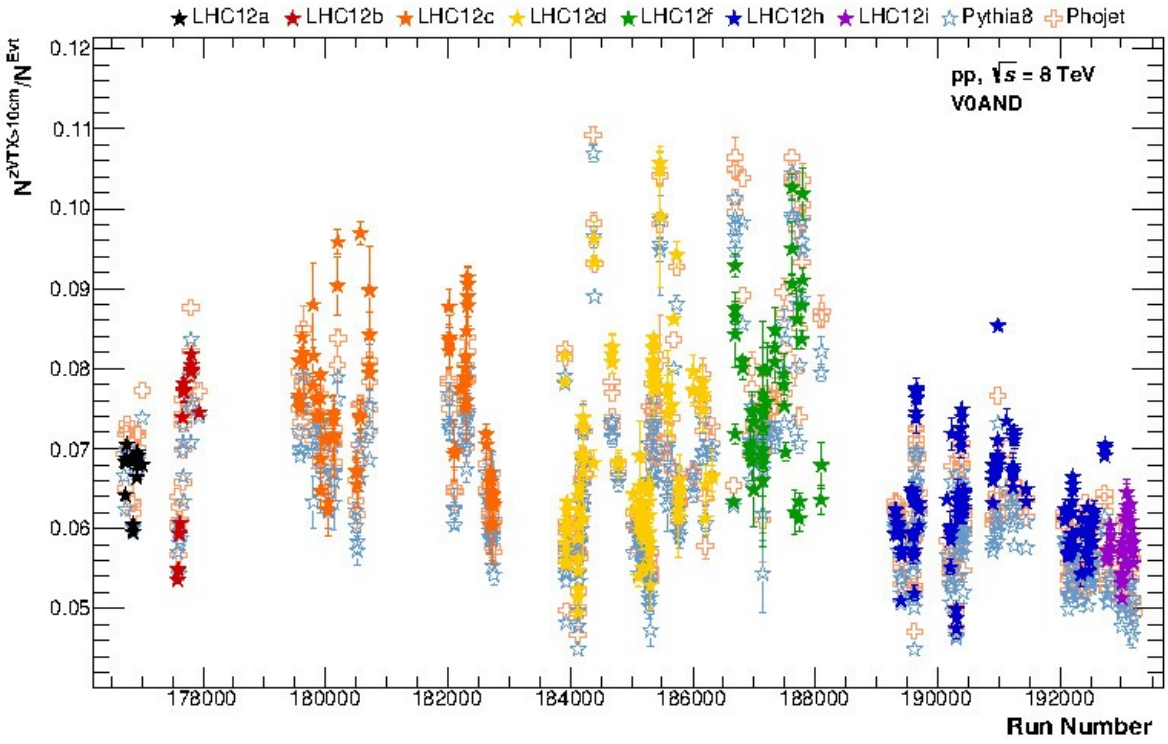
Some example plots for 1.-6.:

1.

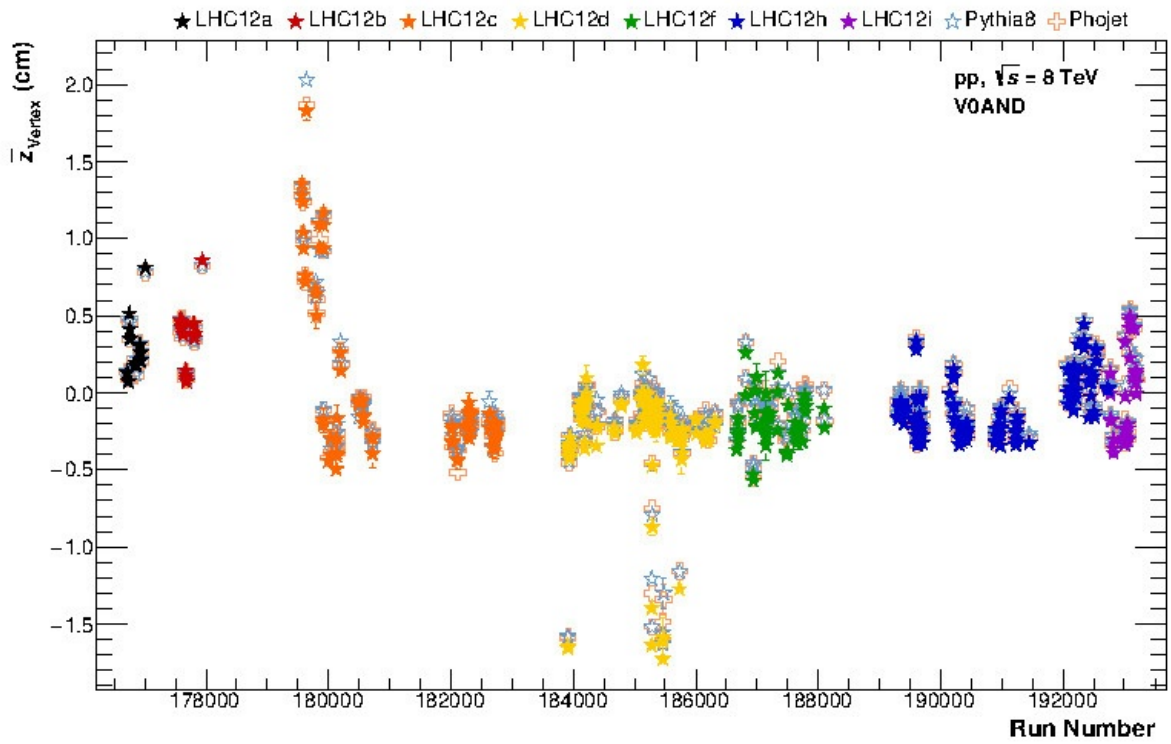



LHC12a LHC12b LHC12c LHC12d LHC12f LHC12h LHC12i Pythia8 Phojet
pp, √s = 8 TeV
V0AND
$\overline{N}_{\text{Good Tracks}}$
8.5
8.0
7.5
7.0
6.5
6.0
5.5
5.0
178000
180000
182000
184000
186000
188000
190000
192000
Run Number

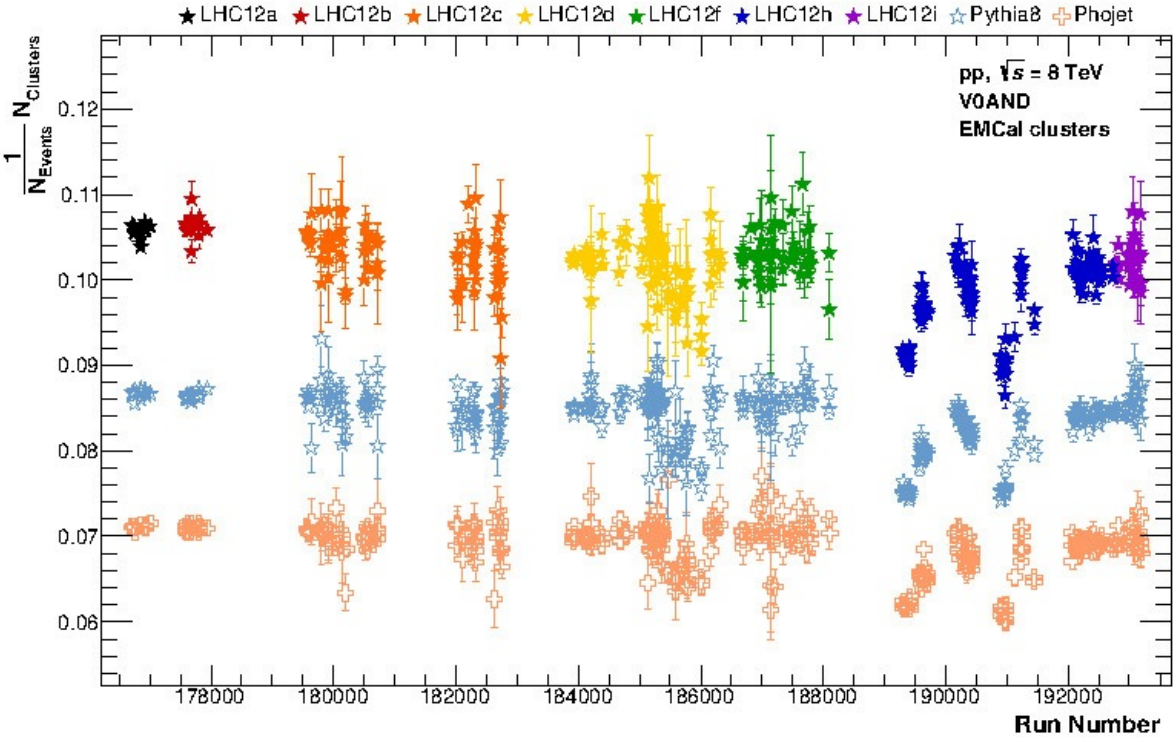
2.


LHC12a LHC12b LHC12c LHC12d LHC12f LHC12h LHC12i Pythia8 Phojet
pp, √s = 8 TeV
V0AND
$N^{zVTX<10cm}/N^{Evt}$
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
178000
180000
182000
184000
186000
188000
190000
192000
Run Number

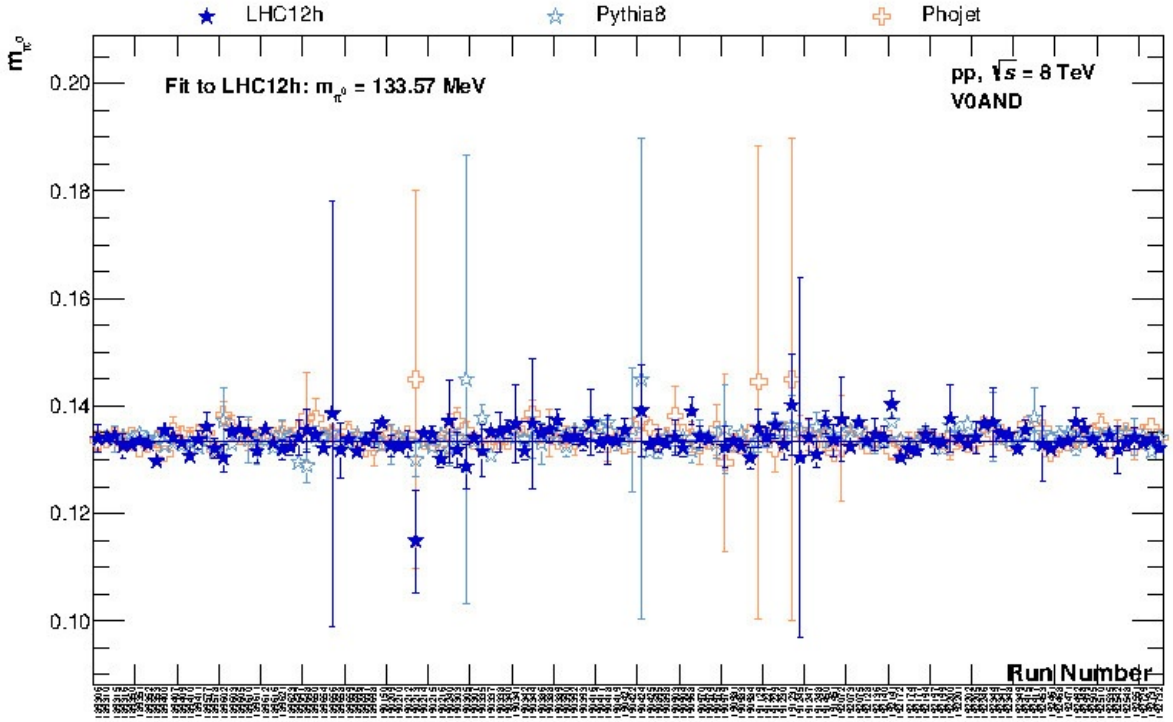
3.


LHC12a LHC12b LHC12c LHC12d LHC12f LHC12h LHC12i Pythia8 Phojet
pp, √s = 8 TeV
V0AND
$\bar{z}_{Vertex}$ (cm)
Run Number

4.


LHC12a LHC12b LHC12c LHC12d LHC12f LHC12h LHC12i Pythia8 Phojet
pp, √s = 8 TeV
V0AND
EMCal clusters
$\frac{1}{N_{Events}} N_{Clusters}$
Run Number

5.


LHC12h
Pythia8
Phojet
m_{π^0}
Fit to LHC12h: m_{π^0} = 133.57 MeV
pp, $\sqrt{s}$ = 8 TeV
V0AND
0.20
0.18
0.16
0.14
0.12
0.10
Run Number

6.

PhotonQA

This part of the QA must be run for any PCM related analysis (for PCM and hybrid analyses PCM-EMCal, PCM-PHOS, PCM-DCal)

The photon QA includes all cut variables in the context of conversion photon analysis:

generated histograms (list of examples) (full set of generated histograms can be deduced from the macros themselves/or from the output generated):

1. Electron level: p_T, η, dE/dx, TPC clusters, findable TPC clusters,...
2. Photon level: p_T, η, ϕ, α, invariant mass, chi2, psi-pair,...

Running the PhotonQA(_Runwise).C will save the output into the following folder structure:

> *CUTNUMBER/SYSTEM/PhotonQA/*

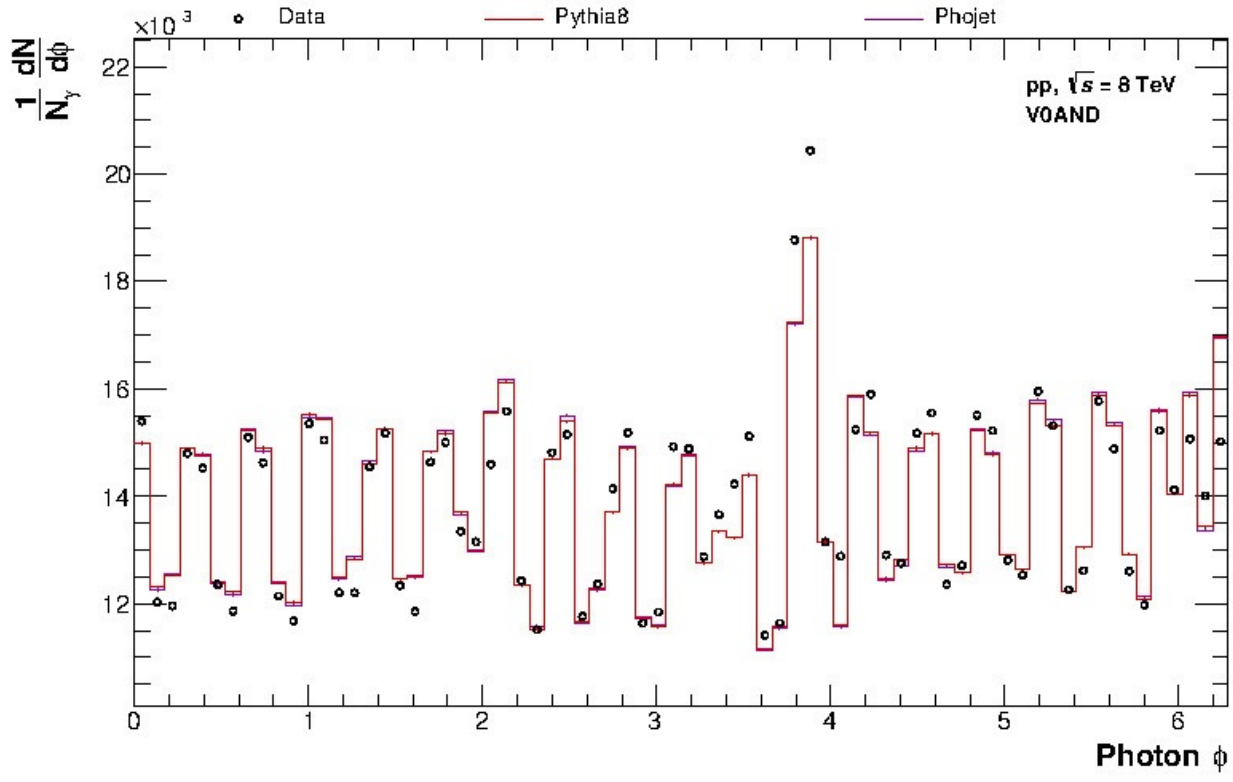
In addition, *.root files will be generated in *CUTNUMBER/SYSTEM/* containing all the histograms as well.

important note Run QA_RunwiseV2.C first(!) with doEventQA = kTRUE and doPhotonQA = kTRUE - it will read TTrees from runwise output and generate runwise QA plots. It will also merge the output from different runs as this is, in general, impossible to do on TTree level due to the huge file sizes. Then run QAV2.C.

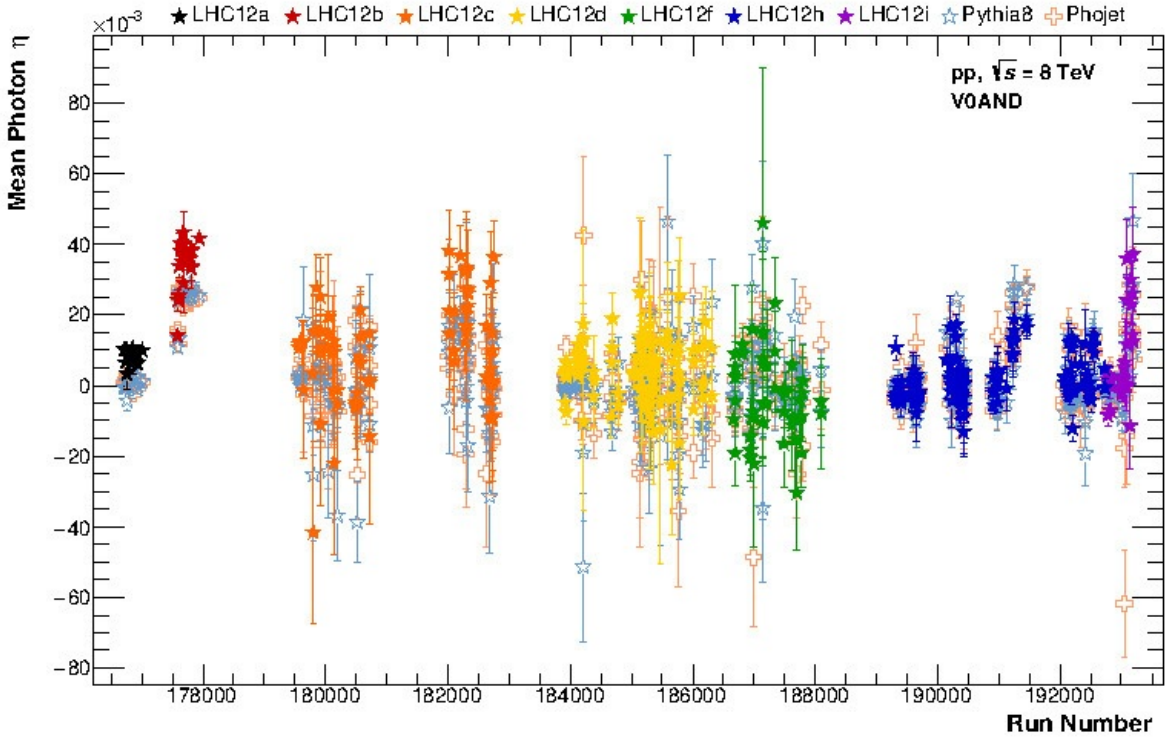
Carefully check all output from runwise histograms with special focus on data/MC comparison (Is the MC able to reproduce all QA histograms extracted from data? Does the MC follow the trends seen in data? Are there any suspicious runs or any observations that cannot be explained?...)

Some example plots for 1. and 2.:

1.



1.


LHC12a
LHC12b
LHC12c
LHC12d
LHC12f
LHC12h
LHC12i
Pythia8
Phojet
pp, $\sqrt{s}$ = 8 TeV
V0AND
Mean Photon η
×10⁻³
Run Number

ClusterQA

This part of the QA must be run for any calorimeter related analysis (for EMCal/PHOS/DCal and hybrid analyses PCM-EMCal, PCM-PHOS, PCM-DCal)

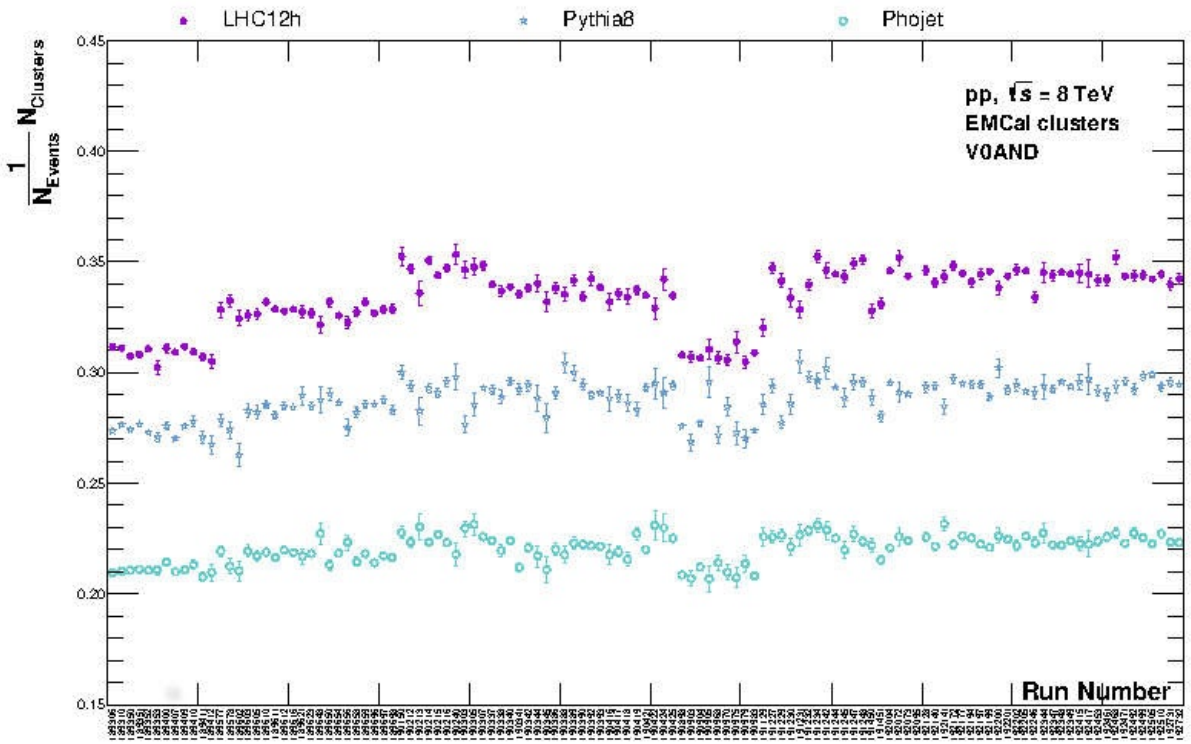
Identify Bad Cells

For bad cell identification, we need the output from GammaCalo with cellQA output contained (perferably option '5') downloaded runwise.

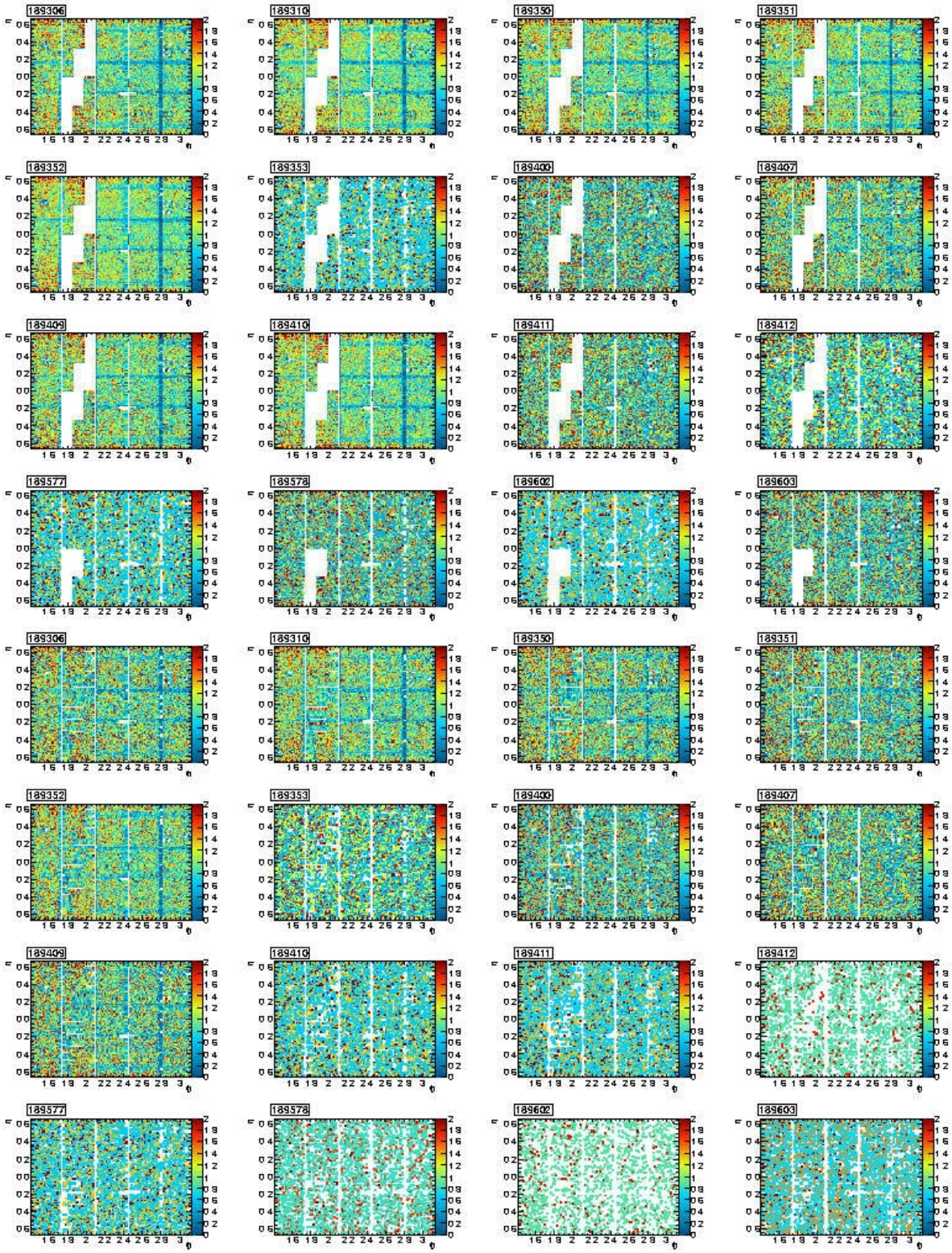
1. Run QA_RunwiseV2.C with doEventQA = kTRUE and doClusterQA = kTRUE (doMergedQA for EMCal merged analysis) with doExtQA == 2! (important for cell level output) using data + MC input in order to generate all needed histograms + txt-files for bad cell identification

Important *example* histograms for the runwise QA and bad cell identification (plots are from the QA stage and include cells which were declared as bad in the meantime):

- Number of Clusters per event



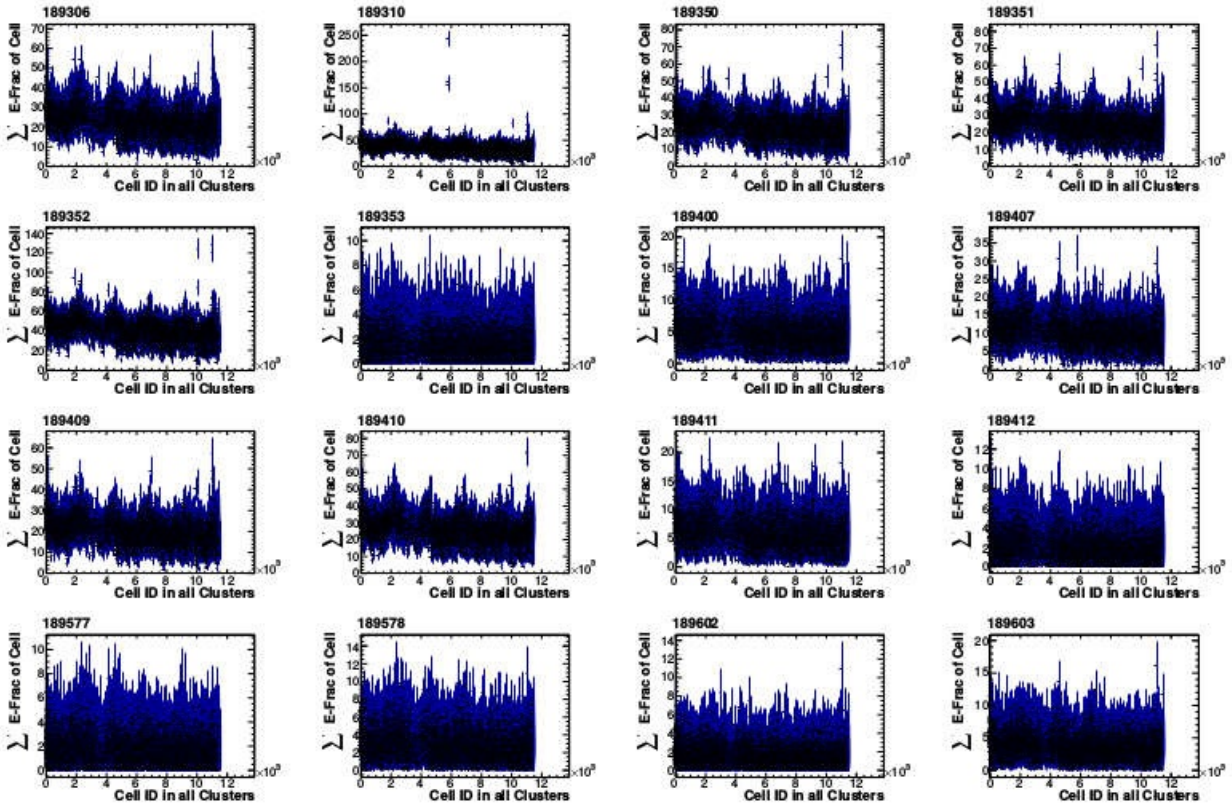
- Cluster eta/phi maps for each run for data and MC - a difference can be clearly seen which **MUST** be corrected in MC (switch off the given RCUs for MC)



2. CellQA step: identify dead and warm/hot cells comparing data and MC information Check histograms created in *ExtQA/** and especially *ExtQA/MissingCells/** for dead cells. Dead/warm/hot cell identification by comparing the cells EFrac to mean EFrac of neighboring cells

> EFrac = cells energy fraction of full cluster energy, summed over all events --> turned out to be a much better discriminator than just looking how often cells fired

An example of the EFrac plotted versus CellID which is generated by the RunwiseQA:



We have a dead cell candidate:

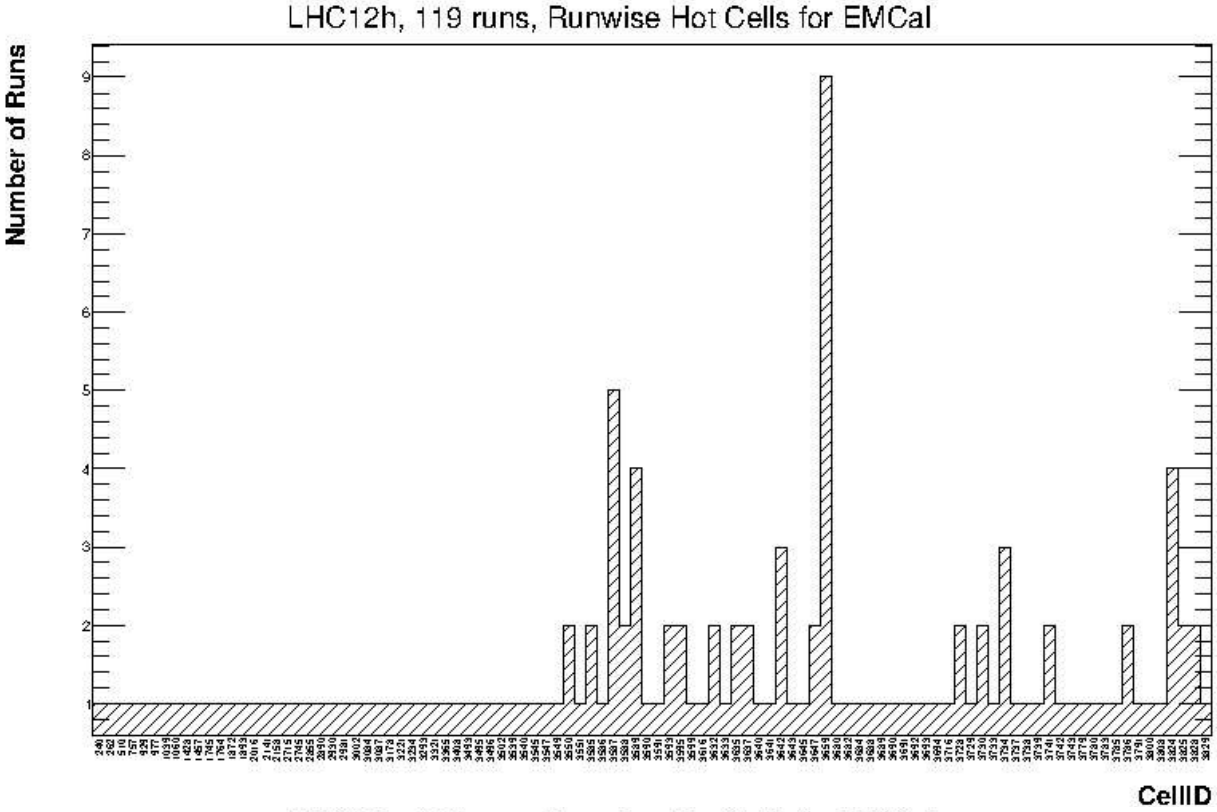
- if((nCurrentEFrac<mean/3 && mean>=80) || (nCurrentEFrac<mean/5 && mean>=40 && mean<80) || (nCurrentEFrac<mean/8 && mean>=10 && mean<40) || (nCurrentEFrac<mean/10 && mean<10))

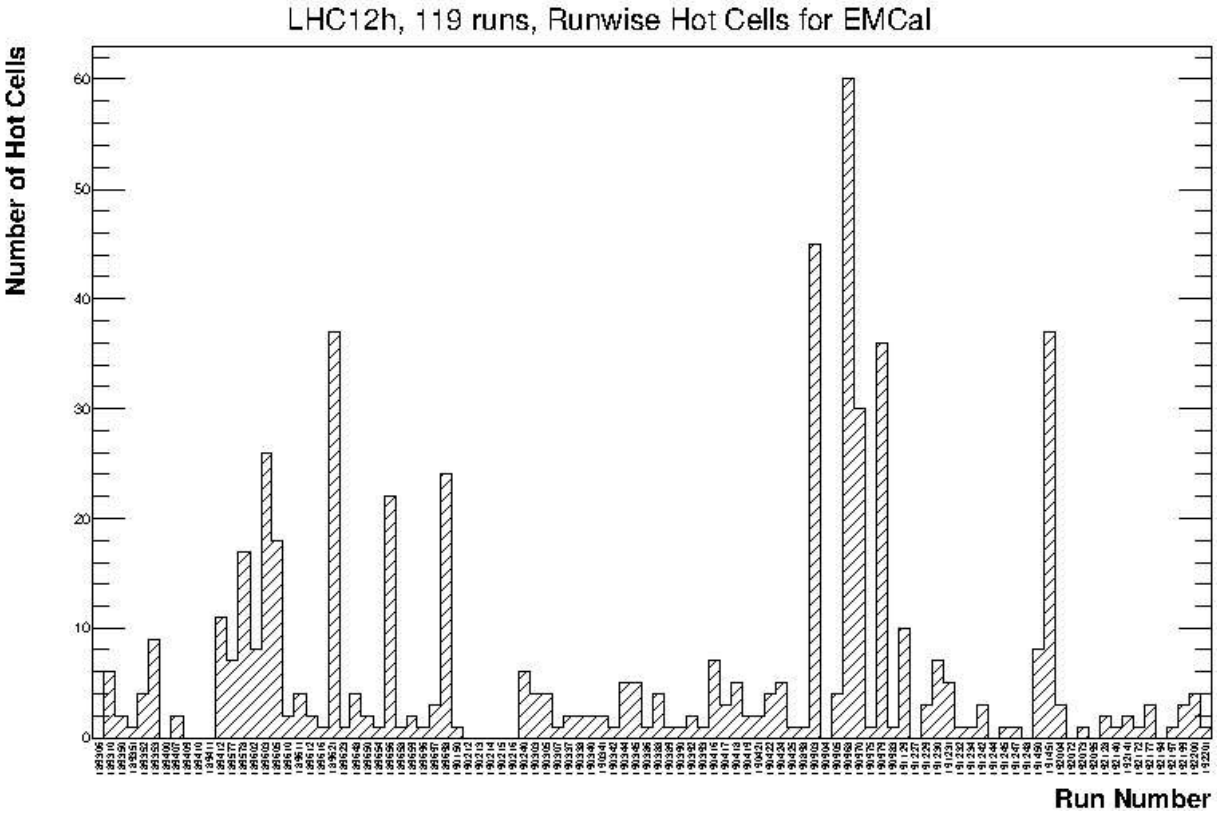
A warm/hot cell candidate:

- if((nCurrentEFrac>2*mean && nCurrentEFrac>80)* || *(nCurrentEFrac>3*mean && nCurrentEFrac>20) || (nCurrentEFrac>4*mean && nCurrentEFrac>8)* || *(nCurrentEFrac>5*mean && nCurrentEFrac>5))

Above conditions for dead/warm/hot cell candidate identification are being applied in function *CheckHotAndColdCellsEFracRunwise* in QA.h! Be careful that a minimum number of events is needed to process the run in this function (runs with ultra low statistics are excluded as it is impossible to look for hot/dead cells besides the obvious candidates which should then appear in other runs as well) The resulting dead/warm/hot cells are written to respective **.log* files that can be found in the output folder.

The hot cell candidates are summarized in such plots (How often does a cell fire per run and how many hot cell candidates are available per run):





Example how the HotCellsRunwise looks like:

```
Run-189306
NoNoisyCellsFound!
Run-189310
1872-CellEFraction:86.2982-CurrentMean:42.4863
5824-CellEFraction:162.839-CurrentMean:34.1267
5825-CellEFraction:155.015-CurrentMean:34.1267
5833-CellEFraction:243.445-CurrentMean:34.1719
10095-CellEFraction:83.0187-CurrentMean:26.2337
11052-CellEFraction:94.292-CurrentMean:36.1453
Run-189350
10095-CellEFraction:52.5966-CurrentMean:17.4002
11052-CellEFraction:71.3287-CurrentMean:22.9911
```

Example how the DeadCellsRunwise looks like:

```
Run-191245
NotEnoughCellsFired-67759-ToObtainColdChannels!
NoColdCellsFound!
Run-191247
NotEnoughCellsFired-87926-ToObtainColdChannels!
NoColdCellsFound!
Run-191248
46-CellEFraction:0.348177-CurrentMean:5.78061
74-CellEFraction:0-CurrentMean:5.78061
103-CellEFraction:0-CurrentMean:5.78061
128-CellEFraction:0-CurrentMean:5.81746
152-CellEFraction:0-CurrentMean:5.72
198-CellEFraction:0-CurrentMean:5.00258
328-CellEFraction:0.23663-CurrentMean:5.02124
353-CellEFraction:0-CurrentMean:5.15569
594-CellEFraction:0-CurrentMean:4.84653
747-CellEFraction:0.390729-CurrentMean:4.39367
759-CellEFraction:0-CurrentMean:4.37241
846-CellEFraction:0.36234-CurrentMean:3.93415
917-CellEFraction:0-CurrentMean:3.63333
1002-CellEFraction:0-CurrentMean:4.16652
1038-CellEFraction:0-CurrentMean:4.18485
1051-CellEFraction:0.346459-CurrentMean:4.21454
```

The next steps 3./4. are needed as by statistical fluctuations some cells can fire more often or less often than usual, therefore we need to apply further conditions for dead/hot cell identification:

3. Run ClusterQA*DeadCellCompare.C (needs to be configured within macro - not _yet* included in steering macros) Determination of dead cells using for-loop around *line 280*. Different decisions when to consider dead cells:

- if cell is dead cell candidate in consecutive number of runs (currently set up with 4),
- if at least 10 dead cell candidates are found with in the same run range or cell candidate is identified to be dead in a given % of analysed runs, represented by fractionThesh.

All results are written/summarized in log-files (examples):

```
CellID:198,cold/notFiredInRuns:176701,176730,176749,176753,176926,176927,176929,177011,177148,177173,177180,177182,
CellID:871,cold/notFiredInRuns:176715,176730,176749,176753,176854,176859,176926,176927,176929,177011,
CellID:1002,cold/notFiredInRuns:176927,
CellID:1306,cold/notFiredInRuns:176854,
CellID:1682,cold/notFiredInRuns:176730,176749,176753,176926,176927,
CellID:1683,cold/notFiredInRuns:176730,176749,176753,
CellID:1735,cold/notFiredInRuns:176927,
CellID:2069,cold/notFiredInRuns:176730,176749,176753,176926,176927,176929,
CellID:2171,cold/notFiredInRuns:176926,
CellID:2179,cold/notFiredInRuns:176730,176749,176753,176926,176927,
CellID:2306,cold/notFiredInRuns:176749,
CellID:2312,cold/notFiredInRuns:176749,177011,
CellID:2313,cold/notFiredInRuns:176715,176730,176749,176753,176854,176924,176926,176927,176929,177011,177167,177182,
```

and according to the selection criteria the final list of dead cells:

```
198 in 70.5882% of selected runs dead/cold
871 in 58.8235% of selected runs dead/cold
2313 in 70.5882% of selected runs dead/cold
2395 in 70.5882% of selected runs dead/cold
2778 in 52.9412% of selected runs dead/cold
3940 in 58.8235% of selected runs dead/cold
10086 in 58.8235% of selected runs dead/cold
11091 in 70.5882% of selected runs dead/cold
```

4. Run ClusterQA*HotCellCompare.C (needs to be configured within macro - not _yet* included in steering macros) Determination of warm/hot cells using for-loop around line 180 via 'threshNFired' and 'threshNTotalFired' for 3. + 4. produce output log-files which summarize the bad cells to be excluded in OADB from runwise dead/warm/hot cell determination.

All hot cell candidates are written/summarized in log-files

```
CellID: 5602, occurred for first time in run: 177180, in total noisy in - 2 - differentRuns, where it occurred in cluste
rs 370.715 times! That is 10.3574 times than the average of neighboring cells!
CellID: 5860, occurred for first time in run: 176730, in total noisy in - 6 - differentRuns, where it occurred in cluste
rs 956.856 times! That is 2.28916 times than the average of neighboring cells!
CellID: 6835, occurred for first time in run: 176730, in total noisy in - 6 - differentRuns, where it occurred in cluste
rs 884.416 times! That is 2.15624 times than the average of neighboring cells!
CellID: 6933, occurred for first time in run: 176730, in total noisy in - 5 - differentRuns, where it occurred in cluste
rs 1038.84 times! That is 2.14903 times than the average of neighboring cells!
CellID: 7104, occurred for first time in run: 176730, in total noisy in - 4 - differentRuns, where it occurred in cluste
rs 851.255 times! That is 2.14486 times than the average of neighboring cells!
CellID: 8420, occurred for first time in run: 176730, in total noisy in - 2 - differentRuns, where it occurred in cluste
rs 299.125 times! That is 2.23993 times than the average of neighboring cells!
```

and according to the selection criteria the final list of hot cells (sorted by run or sorted by cell ID):

(cell ID / run number)

```
11198 176715
11102 176730
11185 176730
11197 176730
11200 176730
11377 176730
5860 176730
6835 176730
6933 176730
7104 176730
8420 176730
8465 176730
8467 176730
8916 176730
9024 176730
9219 176730
9886 176730
10514 176749
9403 176749
11148 176753
9899 176859
5602 177180
```

After identifying dead/hot cells on runwise level, the general periodwise level CellQA + ClusterQA step follows

5. CellQA step on periodwise level via running of ClusterQA.C identify bad cells on periodwise level by using data and MC information.

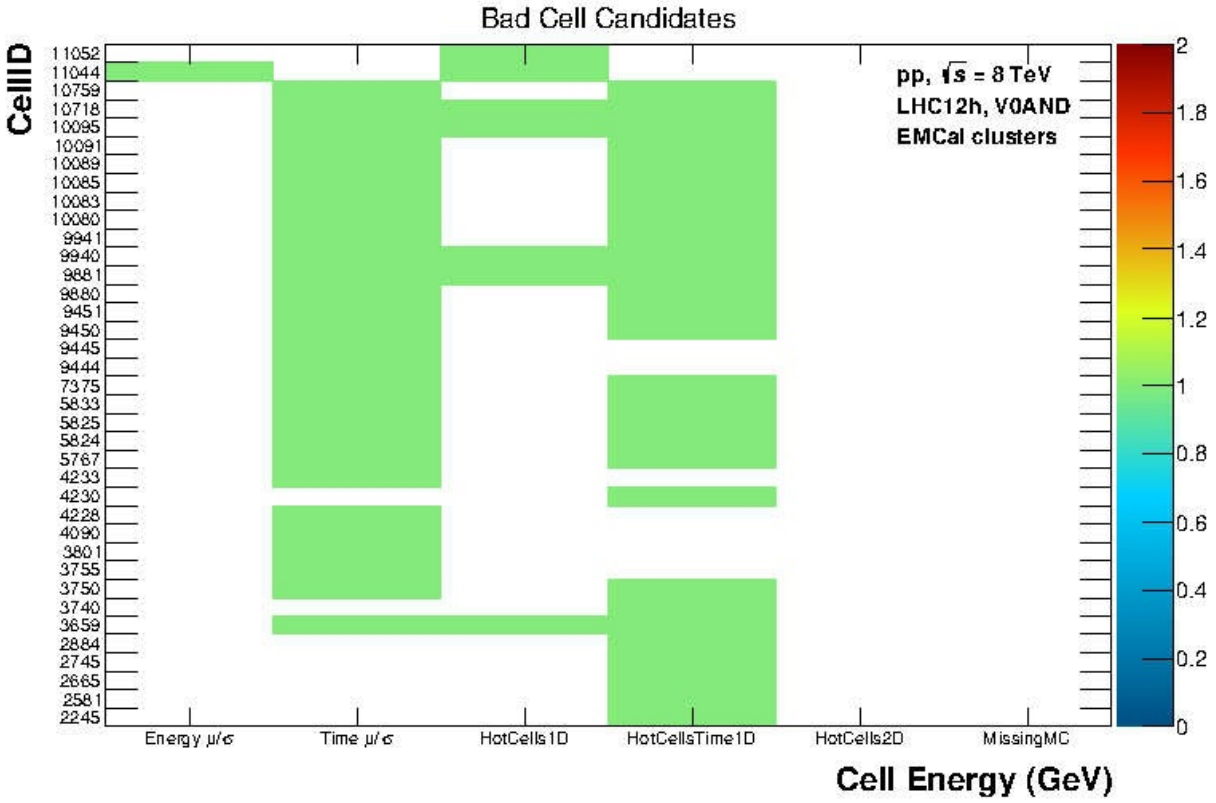
> Setting for bad cell identification are defined in QAV2.C (Double_t arrQAEnergy[4]; Double_t arrQATime[4]; Double_t arrQAHotCells1D[4]; Double_t arrmin2D[9]; Double_t arrmax2D[9];)

The cell IDs are also written to the log-file which is generated:

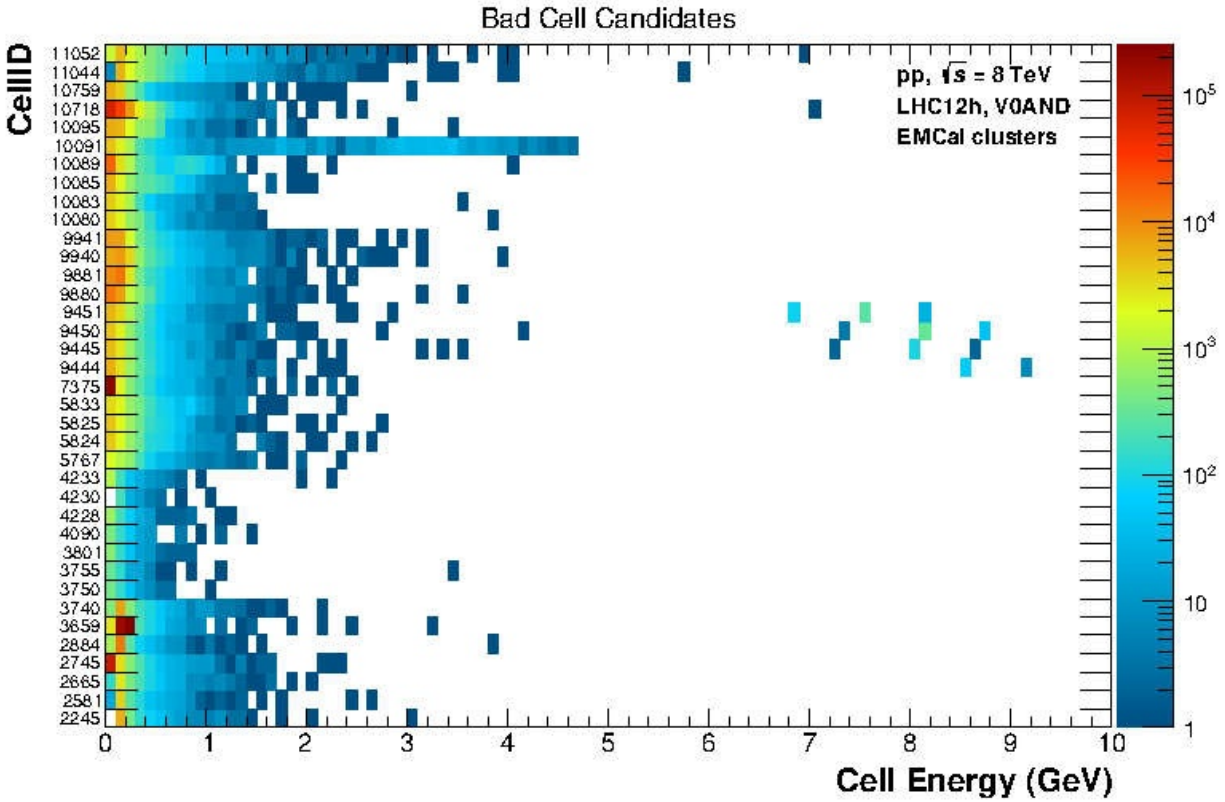
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: Energy - Mean/Sigma
Total number of 1 cells
11044,
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: Time - Mean/Sigma
Total number of 28 cells
3659, 3750, 3755, 3801, 4090, 4228, 4233, 5767, 5824, 5825, 5833, 7375, 9444, 9445, 9450, 9451, 9880, 9881, 9940, 9941,
10080, 10083, 10085, 10089, 10091, 10095, 10718, 10759,
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: HotCells1D
Total number of 7 cells
3659, 9881, 9940, 10095, 10718, 11044, 11052,
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: HotCellsTime1D
Total number of 28 cells
2245, 2581, 2665, 2745, 2884, 3659, 3740, 3750, 4230, 5767, 5824, 5825, 5833, 7375, 9450, 9451, 9880, 9881, 9940, 9941,
10080, 10083, 10085, 10089, 10091, 10095, 10718, 10759,
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: HotCells2D
Total number of 0 cells

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Suspicious cells from: Missing MC-Data
Total number of 0 cells

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
AllCells.size before sort and unique: 64.Finally 37 different cells found!
2245, 2581, 2665, 2745, 2884, 3659, 3740, 3750, 3755, 3801, 4090, 4228, 4230, 4233, 5767, 5824, 5825, 5833, 7375, 9444,
9445, 9450, 9451, 9880, 9881, 9940, 9941, 10080, 10083, 10085, 10089, 10091, 10095, 10718, 10759, 11044, 11052,
```

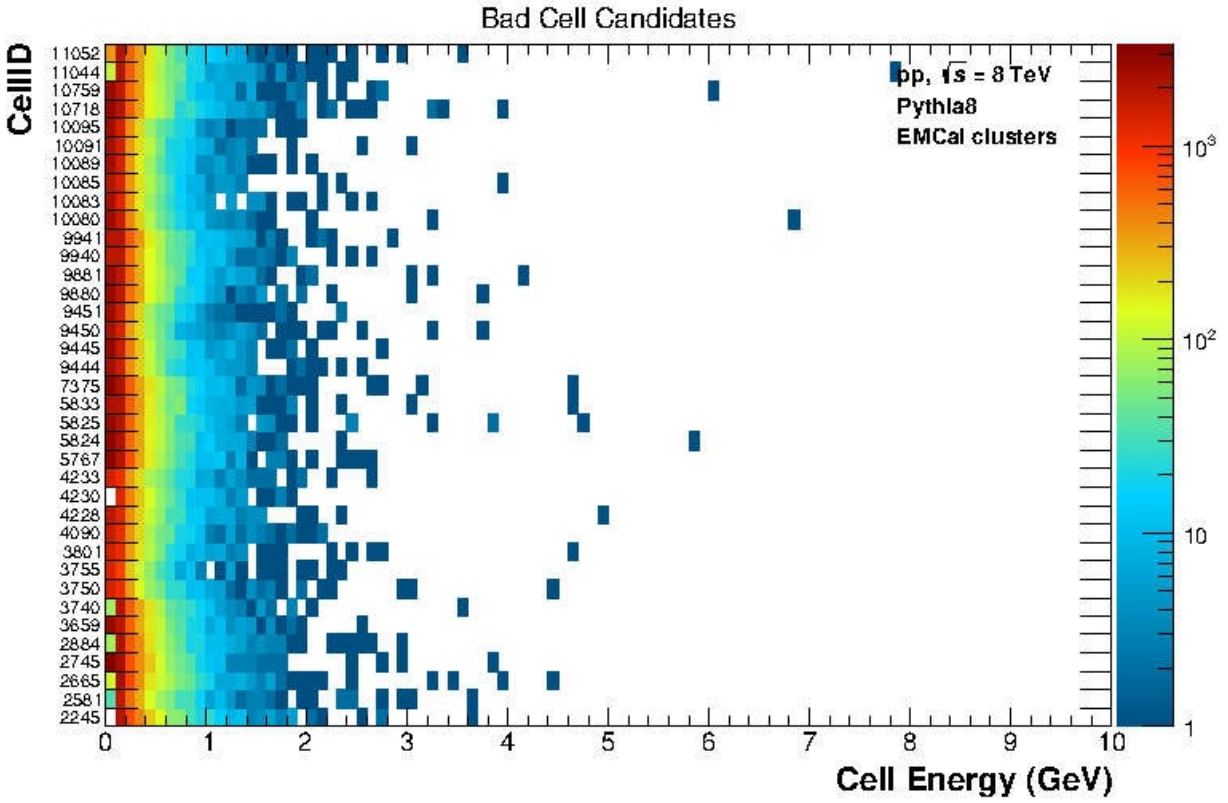
An overview plot is generated with an overview by which cut a cell has been identified as bad candidate (the cell IDs

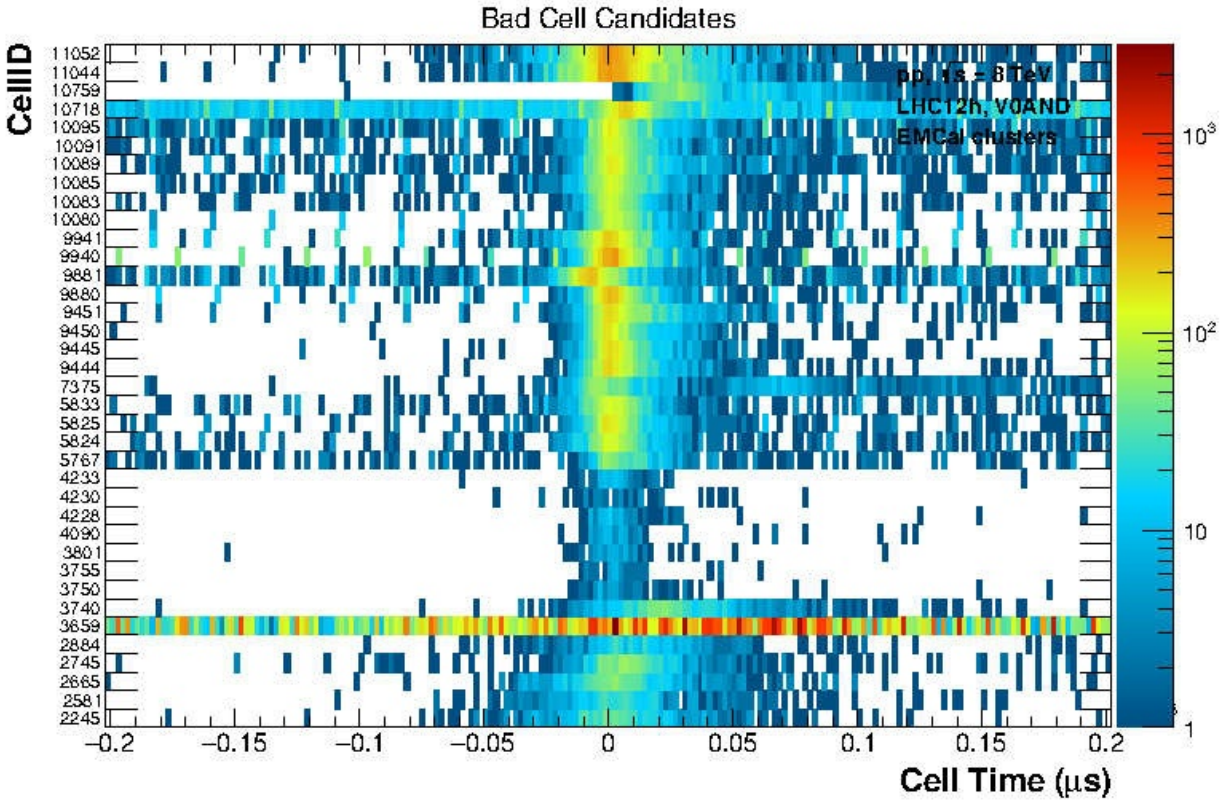
are also stored in log-files):

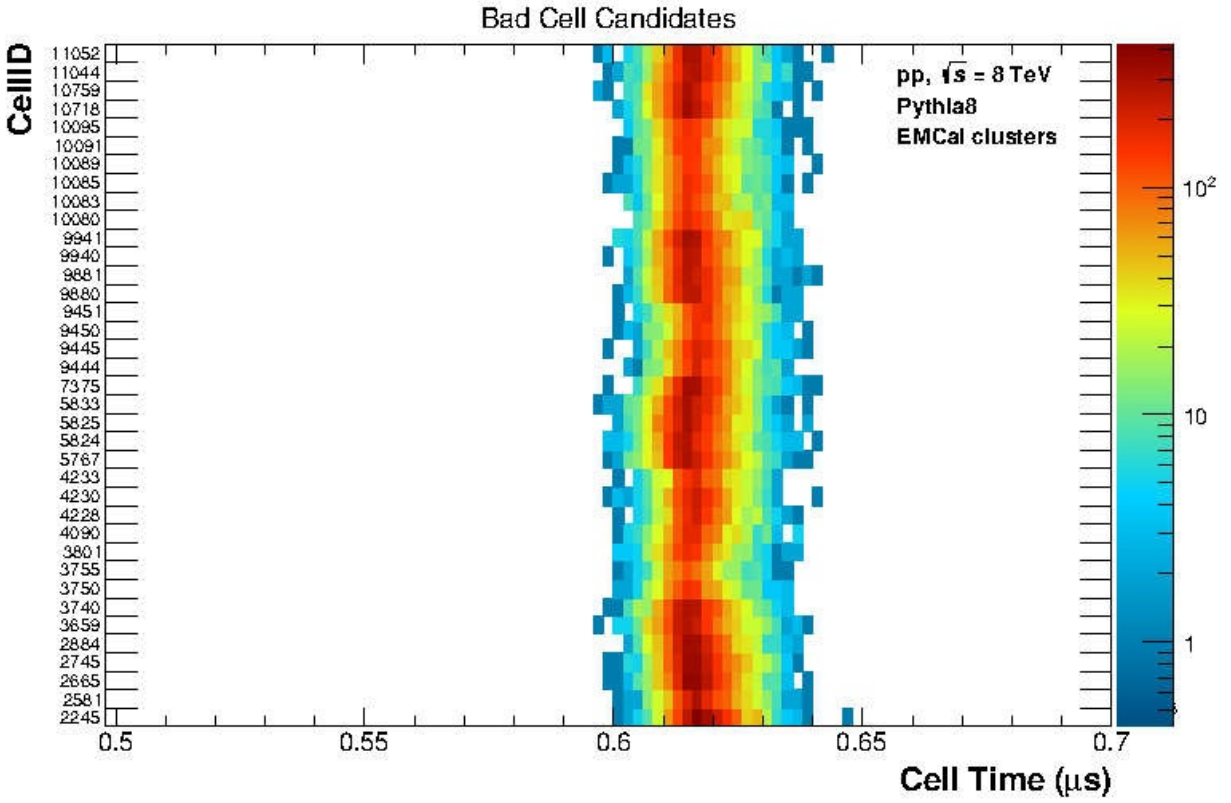


Furthermore, the energy and time distributions are plotted for data and MC:



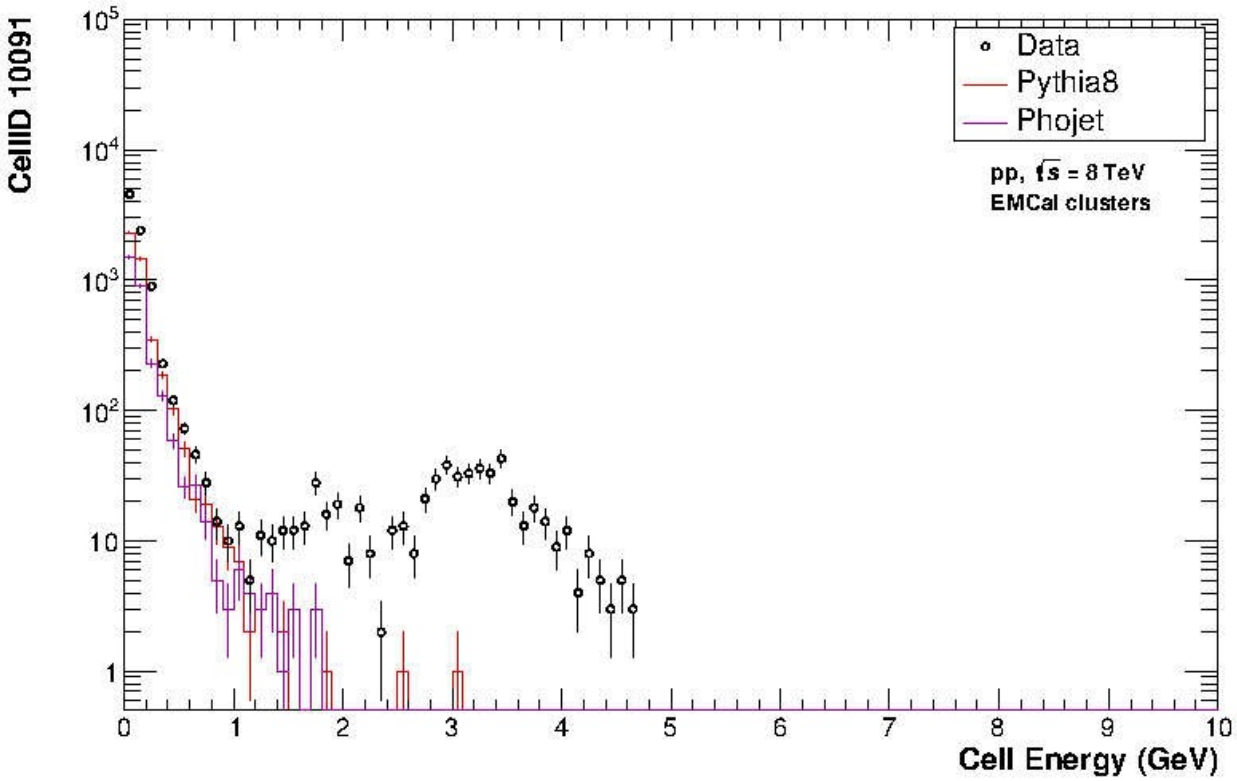

Bad Cell Candidates
CellID
pp, √s = 8 TeV
Pythia8
EMCal clusters
Cell Energy (GeV)


Bad Cell Candidates
CellID
pp, √s = 8 TeV
LHC12h, V0AND
EMCal clusters
Cell Time (μs)



Real bad cell candidates are immediately visible in comparison to the rest of candidates.

In addition, the energy distribution of each cell candidate is compared between data and MC (normalized to the number of events) to help with the judgement if cells are really bad:

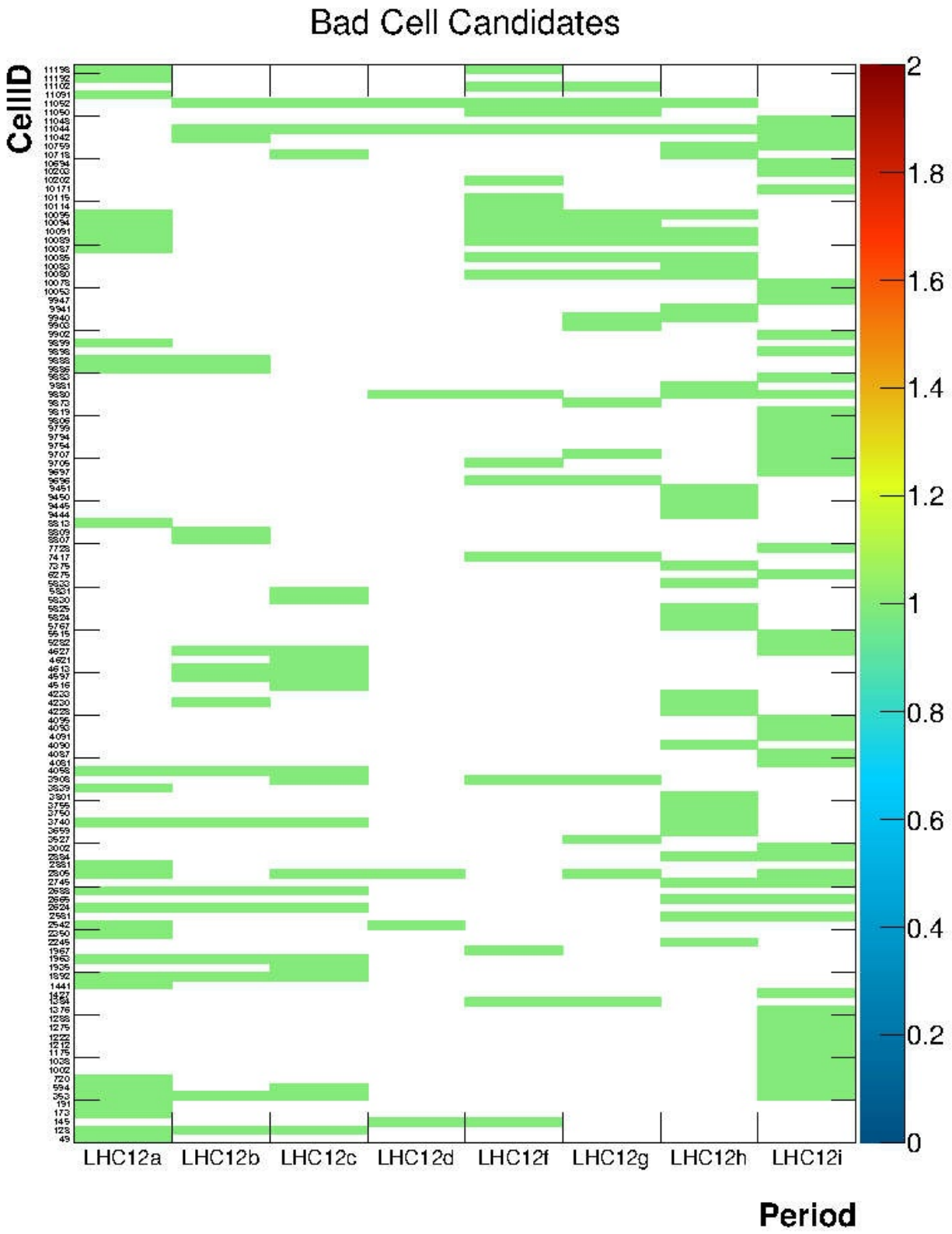


Compare data/MC energy and time distributions for all bad cell candidates found in 5. Also use 8.

Then, merge bad cell candidate list from 5. with lists found in 3. and 4. to be excluded in OADB

6. Run ClusterQA*CellCompare (needs to be configured within macro - not _yet* included in steering macros) to visualize bad cell candidates found in step 5.

An overview plot is generated with an full overview in which period a cell has been identified as bad candidate:



Run ClusterQA

The cluster QA covers general cluster+cell properties like:

generated histograms (list of examples) (full set of generated histograms can be deduced from the macros themselves/or from the output generated):

- Cut overview histograms (vs. p_T)
- Cell histograms (number of cells fired, cell energy/timing,...)
- Cluster histograms (number of clusters per event, eta/pi distributions,...)

Running the ClusterQA(_Runwise).C will save the output into the following folder structure:

> *CUTNUMBER/SYSTEM/EventQA/*

In addition, *.root files will be generated in *CUTNUMBER/SYSTEM/* containing all the histograms as well.

7. Runwise ClusterQA step by analysing output from ClusterQA_Runwise.C Carefully check all output from runwise histograms with special focus on data/MC comparison (Is the MC able to reproduce all QA histograms extracted from data? Does the MC follow the trends seen in data? Are there any suspicious runs or any observations that cannot be explained?...)

8. ClusterQA step by analysing output from ClusterQA.C Carefully check all output from histograms with special focus on data/MC comparison (Is the MC able to reproduce all QA histograms extracted from data? Does the MC follow the distributions seen in data? Are there any suspicious observations or is there anything that cannot be explained?...)

9. Optional: run ClusterQA*Compare (needs to be configured within macro - not _yet* included in steering macros) to compare different data sets (MB vs calorimeter trigger for example for a given dataset; needs input from runwiseQA and periodwiseQA)

Carefully check all output from runwise/full output with special focus on data/MC comparison (Is the MC able to reproduce all QA histograms extracted from data? Does the MC follow the trends seen in data? Are there any suspicious runs or any observations that cannot be explained?...) In general, they should be stable vs. run number - however, one of the exceptions is pileup which may vary from run to run -> need to be taken with special care!

Debug Mode / Visualise Clusters

There is an existing option to write specific clusters to a txt-file with the *GammaCalo* task (for example cluster combinations with a very small opening angle or select especially huge clusters,...). Once you produced the txt-files with help of the *GammaCalo* task, you can visualize the clusters by help of the macro *Grid_PlotGammaCaloDebug* which can be found in *TaskQA*.

It can be called via:

> Grid_PlotGammaCaloDebug(TString filePath = "/home/daniel/data/work/debugOutput.txt", TString outputDir = "DebugPlots", Bool_t plotFullEMCal = kFALSE, Int_t debugOption = 0)

PrimaryTrackQA

This part of the QA only must be run when performing a neutral pion analysis via the *Dalitz decay channel* (γe^+e^-) or an ω analysis.

To-Do (Jens)

Energy Calibration of Calorimeters

A detailed explanation of the procedure can be found in Analysis Notes EMCal @ pp 8 TeV, PCM-EMCal @ pp 8 TeV as well as EMCal @ pp 2.76 TeV and PCM-EMCal @ pp 2.76 TeV.

The macros needed to perform the energy calibration can be found in *AnalysisSoftware/TaskV1*, which are **CorrectCaloNonLinearityV4** and **CorrectCaloNonLinearityV4_Compare**. They can be run using:

```
root -x -l -b -q 'TaskV1/CorrectCaloNonLinearityV4.C+("config.txt", "eps", kFALSE)'

root -x -l -b -q 'TaskV1/CorrectCaloNonLinearityV4_Compare.C+("config.txt", "eps", kFALSE)'
```

The first argument is the config-file, for which example configs can be found in *TaskV1/ExampleConfigs*. Then the output format of the figures is given as well as a flag to give further debug output with cout's.

Important things to look for:

- Is the background subtraction working properly?
- Cross-check the fits: do they all converge?
- Are the transverse momentum bins defined properly? If you have massive statistics, split them. Otherwise merge bins. Especially check high momentum bins, if signal extraction is viable.
- Are triggers available for the data? If yes, use them as they will give you much better handle on the higher momenta.
- Compare the different calibrations, do they make sense? In principle, the different procedures CMF, CRF, CCMF, CCRF (-> see analysis notes for explanations) etc should give very similar results.

Once you obtained the correction function, it must be implemented in AliPhysics into the helper task *AliCaloPhotonCuts*.

- Check, if the MC you are working on is already implemented in the framework.
- Add your calibration to the function *ApplyNonLinearity* in *AliCaloPhotonCuts* for the MC you are working on.
- Check, if you have the correct cut-numbers available in your AddTasks to run the LEGO trains on the GRID -> if not, add the cut configs.
- Do the commit and request the LEGO train using your calibration.

IMPORTANT NOTE

After applying your calibration, rerun your analysis and use your newly acquired calibration. Then repeat the calibration step explained above and verify that mass ratios are at '1'. If this is not the case, something went wrong and you need to look for the issue.

Photon, Electron, and Dielectron Cocktails in ALICE

Organization

e-group: *alice-cocktail-EM* **twiki**: EMCocktail **contacts**:

- photons: Friederike Bock friederike.bock@cern.ch
- electrons: Ralf Averbeck ralf.averbeck@cern.ch
- dielectrons: Oton Vazquez Doce oton.vazquez@universe-cluster.de

Software repository for parametrization & spectra input

repository: **Cocktail-Repository**
in order to check it out first go to the web page and add your ssh key. readme
afterwards you can check it out with:

```
git clone ssh://git@gitlab.cern.ch:7999/alice-cocktail-EM/cocktail_input.git
```

If you can't access, please inform Friederike Bock

This repository is meant to collect all the available spectra and v_n within ALICE for the different particle species ($\pi^{0,\pm}, \eta, \omega, K^{0,\pm}$ ...) which are needed to create the electromagnetic cocktails for the photon, heavy flavour and di-electron analysis. As such the information from different analyses has been summarized in 3 different twiki's and the corresponding macros to compile the data including the links to the analysis notes and JIRA tickets for the data which are not yet published. These are for:

- **pp**: CocktailProduceCompleteInputFilePP.C with the corresponding PP-twiki
- **p-Pb**: CocktailProduceCompleteInputFilePPb.C with the corresponding PPb-twiki
- **Pb-Pb**: CocktailProduceCompleteInputFilePbPb.C with the corresponding PbPb-twiki

As these pages and macros are maintained volluntarily please keep in mind that they might not reflect the latest status. Thus you should feel free to update them if you find a link or result does not correspond any longer to the knowledge within ALICE. We ask however that you make it available to all of us and update it in both places the macro and the twiki, so that we keep them in sync.

The latest fully vetted version of the code is usually kept in the *master-branch* of the cocktail-repository. For the latest changes you might, however need to check out the *devel-branch* of the same directory. If necessary we can create a separate branch for your changes as only a few people can directly commit to the *master-branch* in order to allow a full vetting of the changes.

How to make a cocktail simulation

The generator level particle decay simulation, also called "cocktail simulation" is used for the secondary neutral pion or secondary photon correction as well as for the decay photon spectra necessary for a direct photon measurement.

The cocktail itself is based on parameterizations of (measured) particle spectra and uses mT scaling for the remaining spectra. In the following the basic steps for the cocktail generation are described exemplary for pp 5 TeV. This includes the procedure to add new input spectra to the cocktail, the parametrization of these spectra and finally the generation of the simulation.

The cocktail framework repository can be found at **https://gitlab.cern.ch/alice-cocktail-EM/cocktail_input**. The most updated branch is the "devel" branch which is merged into the master on a monthly basis. It is therefore recommended to checkout the devel branch when working on the cocktail framework.

Adding new input spectra

In this part of the tutorial it is explained how to add new spectra to the cocktail. This is done in the macros **CocktailProduceCompleteInputFilePP.C, CocktailProduceCompleteInputFilePPb.C or CocktailProduceCompleteInputFilePbPb.C** depending on the desired collision system. *Side remark: XeXe is not yet included in the framework*. These macros are controlled with the respective bash scripts (e.g.via **script_startProduce_pp.sh newInputPP**). This will create the input spectra file for all currently defined center of mass energies from 0.9 to 13 TeV.

In order to add spectra to the pp 5 TeV cocktail it is necessary to have the `TList *list_5TeV` defined and activated via the first argument of the macro. This first argument `TString enableEnergy = "111111"` is a simple switch for all defined center of mass energies starting from the left with 0.9 TeV and the last digit being the switch for 13 TeV. For 5 TeV, at least the third digit in the string should be set to 1 (e.g. "001000"). The lists for the different center of mass energies are filled in ascending energy order starting with 900 GeV. The 5 TeV spectra are added within the `if (Include_5TeV) { ... }` condition.

In addition to the center of mass energy switch, also **each particle has its own boolean switch** which are defined in the macro arguments (e.g. `enable_NPion` , `enable_Eta` , ...). To now add the neutral pion spectra to the list of input spectra, the condition `if (enable_NPion ) { ... }` or `if (enableNPion || enableEta ) { ... }` is used. Usually the latter is used as the neutral meson spectra are measured by us and provided in the file format produced by ProduceFinalResultsTriggersPatched/CombineMesonMeasurements as explained in the neutral meson section including systematic uncertainties.

The file containing the spectra then needs to be loaded and also added to the respective directory in the repository (pp, pPb, PbPb). The spectra can be loaded as `TH1D, TGraphErrors or TGraphAsymmErrors` whereas the latter usually allows for the best/easiest fitting.

The spectra then need to be added to the list as **invariant cross section** in the condition `if (changeQuantity == 0) { ... }` and as **invariant yield per inelastic event** otherwise. This requires the right multiplication/division of the respective cross sections to result in the correct quantity. (e.g. our cross sections are either V0OR or V0AND and therefore need to be divided by the inelastic cross section to obtain invariant yield per inelastic event). Before the spectra are added to the list, they also **need to be converted to dN/dydpT** with the functions `SetGraphProperties(...)` , or `SetHistoProperties(...)` depending on the format. This also creates the necessary naming scheme of the spectra (e.g. NPionPCMStat, EtaCombSys, ...). For this, the right `fParticle[]` and `fMethod[]` array entries need to be set in the `SetGraph/HistoProperties(...)` function which can be found in CocktailFunctions.h for all particles and methods (e.g. neutral pions are `fParticle[0]` and the EMC method is `fMethod[4]`).

To summarize:

1. Create list for given center of mass energy (if not yet defined)
2. Load input spectra (TH1D, TGraphErrors or TGraphAsymmErrors) with separate spectra for statistical and systematic uncertainties
3. Convert spectra to invariant yield per inelastic event and afterwards to dN/dydpT
4. Apply correct naming scheme using the arrays fParticle[] and fMethod[]
5. Add spectra to the list

In the following, this part is shown in the full code implementation for a combined neutral pion spectrum at 5 TeV.

```
if (Include_5TeV){
        if (enable_NPion || enable_Eta){
            TString localOutputQuantity                             = "#it{E} #frac{d^{3}#sigma}{d#it{p}^{3}} (pb GeV^{-
2} #it{c}^{3})";

            TFile* fFileNeutralMeson5TeVComb                        = new TFile("pp/CombinedResultsPaperPP5TeV_2017_10_1
1.root");
            TDirectoryFile* fFileNeutralMeson5TeVCombPi0            = (TDirectoryFile*)fFileNeutralMeson5TeVComb->Get("P
i05TeV");
            TDirectoryFile* fFileNeutralMeson5TeVCombEta            = (TDirectoryFile*)fFileNeutralMeson5TeVComb->Get("E
ta5TeV");

            if (enable_NPion){
                TGraphAsymmErrors* graphNPionXSecEMCAL5TeVStat      = (TGraphAsymmErrors*)fFileNeutralMeson5TeVCombPi0->
Get("graphInvCrossSectionPi0EMCAL5TeVStatErr");
                TGraphAsymmErrors* graphNPionXSecEMCAL5TeVSys       = (TGraphAsymmErrors*)fFileNeutralMeson5TeVCombPi0->
Get("graphInvCrossSectionPi0EMCAL5TeVSysErr");

                if (changeQuantity == 0){
                    SetGraphProperties(graphNPionXSecComb5TeVStat,  Form("Xsec_%s%sStat",    fParticle[0].Data(), fMetho
d[1].Data()), "#it{p}_{T} (GeV/#it{c})", localOutputQuantity, "");
                    SetGraphProperties(graphNPionXSecComb5TeVSys,   Form("Xsec_%s%sSys",     fParticle[0].Data(), fMetho
d[1].Data()), "#it{p}_{T} (GeV/#it{c})", localOutputQuantity, "");

                    list_5TeV->Add(graphNPionXSecComb5TeVStat);
                    list_5TeV->Add(graphNPionXSecComb5TeVSys);

                } else {
                    // convert XSection to invariant yield
                    TGraphAsymmErrors* graphNPionComb5TeVStat       = ScaleGraph(graphNPionXSecComb5TeVStat,1./xSection5
023GeVINEL);
                    TGraphAsymmErrors* graphNPionComb5TeVSys        = ScaleGraph(graphNPionXSecComb5TeVSys,1./xSection50
23GeVINEL);

                    // convert inv yield to 1/N d^2N/dydpT
                    graphNPionComb5TeVStat                          = ConvertYieldGraph(graphNPionComb5TeVStat,  kFALSE,
 kFALSE, kTRUE, kTRUE);
                    graphNPionComb5TeVSys                           = ConvertYieldGraph(graphNPionComb5TeVSys,   kFALSE,
 kFALSE, kTRUE, kTRUE);

                    SetGraphProperties(graphNPionComb5TeVStat,      Form("%s%sStat", fParticle[0].Data(), fMethod[1].Dat
a()), "#it{p}_{T} (GeV/#it{c})", globalOutputQuantity, "");
                    SetGraphProperties(graphNPionComb5TeVSys,       Form("%s%sSys",  fParticle[0].Data(), fMethod[1].Dat
a()), "#it{p}_{T} (GeV/#it{c})", globalOutputQuantity, "");

                    list_5TeV->Add(graphNPionComb5TeVStat);
                    list_5TeV->Add(graphNPionComb5TeVSys);
                }
            }
            if (enable_Eta){
                //...
            }
      }
}
```

This procedure can be repeated for all available particle spectra. However, sometimes one might only have ratios of particle spectra available or prefer to use the ratios for the parametrization. The only difference is then that there is, of course, no normalization to inelastic events necessary and that the SetGraph/HistoProperties(...) arguments are as follows:

```
SetGraphProperties(graphEtaToPi0CombStat, Form("%sTo%s%sStat", fParticle[1].Data(), fParticle[0].Data(), fMethod[1].Data()), "#it{p}_{T} (GeV/#it{c})", "#eta/#pi^{0}", "");
```

Where the `fParticle[]` array entries for both particles of the ratio are required for the naming scheme. Usually, we use the eta/pi0 ratio for the parameterization of the eta spectrum to have a stable high pT tail compared to the neutral pion. It is therefore recommended to always add the ratio in addition to the eta spectrum itself to the list.

Parametrization

Once all spectra are added and the input spectra file is created (CocktailInputPP.root), the spectra can be parametrized. The fitting is done with **CocktailInputParametrization.C** in combination with three text files as input. As given in the bash script, the macro is called the following way.

```
root -x -l -b -q 'CocktailInputParametrization.C+("CocktailInputPP.root","pp_5TeV","eps","parametrizationSettings/pp_5TeV_standard.dat","parametrizationSettings/pp_5TeV_ratio_standard.dat","parametrizationSettings/listAllUniqueNames_5TeV.txt",kFALSE,kFALSE,0,kFALSE)'
```

With the input spectra file as first argument, the name of the list as second argument, the figure file format as the third followed by the three text files. The macro itself does not need to be modified for this step. **The only way to influence/change the fitting is via the first text file** (e.g. parametrizationSettings/pp_5TeV_standard.dat). This file has to be created if it does not already exist and is setup the following way:

```
%particle centrality method pt const relsys(%) ptMin ptMax func opt par1 par1Low par1Up par2 par2Low par2Up par3 par3Low par3Up par4 par4Low par4Up
```

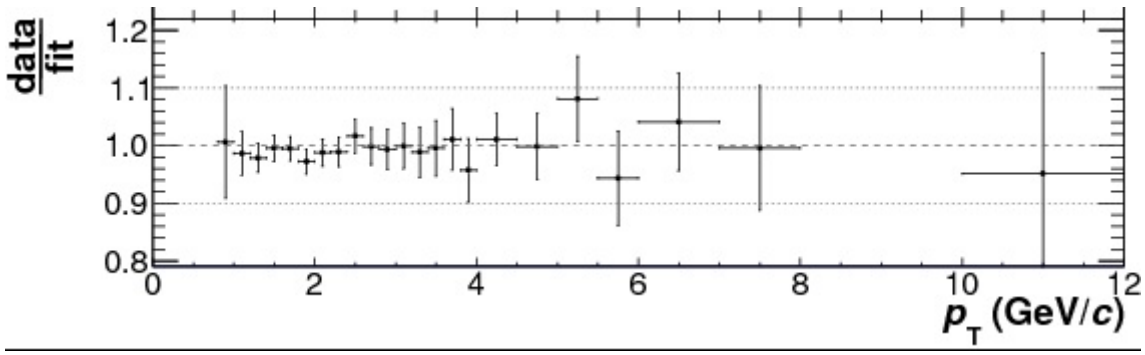
```
NPion MB EMCal 0.042 0 14 oHagPt QNRME+ 511.782 - - 0.1522 - - -0.13 - - 0.44 - - 6 - - stop
```

```
Eta MB PCMEMCal 0.045 1.2 25 tmpt QNRME+ stop
```

Each line represents one parametrization that can be used in the cocktail. In this case we have the neutral pion measurement from EMC and the eta meson with PCMEMC. The fourth argument is the portion of the systematic uncertainty that is flat in pT (e.g. 9% for PCM from the material budget or 4.2% for EMC). This is followed by the pT range that should be used to fit the spectra. Here 0--14 GeV/c for the pi0 and 1.2--25 GeV/c for eta. For fitting itself a variety of fitting function is available, ranging from Levy-Tsallis over Modified Hagedorn to TCM fits multiplied by pT. A full overview of these functions can be found in CocktailFitting.h::FitObject(). **As all of the input spectra are multiplied by pT, the fitting functions should also be the respective functions*pT (e.g. `oHagPt` instead of `oHag`).** The next argument is parsed to the fitter and should generally always be QNRME+. If a fit does not need finetuning, the line can then be ended directly via `stop` . It is, however, also possible to set starting parameters for the fits as well as upper and lower parameter limits. Many examples for this can be found in the folder parametrizationSettings in the repository.

The fitting can then be checked in the plots produced by the macro. They can be found in plots/eps/pp_5TeV in this case. Of interest are here first of all the plots with the name spectra_NPionEMCal_default.eps and spectra_EtaPCMEMCal_default.eps as these show the ratio of the fit that was defined in the text file above to the spectrum. The fitting parameters/functions/ranges need to be optimized to archieve a good description of the spectra over the full pT range. The high pT region needs to be checked thoroughly as the fit functions will be extrapolated up to 200 GeV/c for the cocktail production.At low pT it is important to not drastically overshoot or undershoot the data and to have the fit function go through (x=0, y=0).

The figure below shows a typical ratio of data to fit with a good description from low up to high momentum. Due to statistical fluctuations it can also happen that a datapoint is outside of the +-20% range of the ratio plot. In this case it is important to check the high momentum tail of the fit thoroughly.



Once all spectra fits that are defined in ,e.g. parametrizationSettings/pp_5TeV_standard.dat, are made, the fitting procedure goes towards ratio fits. These are defined in this case in parametrizationSettings/pp_5TeV_ratio_standard.dat. This file is build similar to the non-ratio fit file and plotted figures can be found in the same folder as before. In case of the ratio fits it is important to make sure they cross (0,0) and that they flatten out towards high momentum.

In order to check the systematic variations of the cocktail, which are obtained via shifitng of the datapoints based on the systematic uncertainties in either all points up or down, two linear variation as well as polinomial variations as well as variations of the mT scaling factors. These variations can be checked in the plot spectra_NPionEMCal_paramAllToCentral.eps where the variations should produce rather symmetrical patterns around unity.

Once all spectra are parametrized, the macro will save sets of those parametrizations to the output file that will later on be fed into the cocktail. These sets can be setup for example via parametrizationSettings/pp_5TeV_EMCal_cocktail_settings.dat which is written the following way:

```
%pdg code mtScaleFactor mtScaleFacUp mtScaleFacDown particle method ratio method
111 - - - NPion EMCal stop
221 - - - NPion EMCal EtaToNPion Comb stop
443 0.054 0.056 0.052 stop
```

Each line, again, represents one particle that should be included in the set. The line starts with the PDG code of the particle (e.g. 111 for the neutral pion). The next three entries represent the mT scaling factors for particles that were not parametrized in the previous step. Here, particle 443 is mT scaled with a factor of 0.054 from the neutral pion. The next entries are self explanatory, as we in this case want to use the neutral pion parametrization from the EMC method. For the eta meson, we want to use the eta/pi0 ratio from the combined measurement multiplied with the pi0 EMC parametrization. This provides a very nicely described eta parametrization that keeps the high pT ratio value between pi0 and eta.

The resulting sets of parametrizations are plotted again in the folder parametrizations where also the .root file can be found. This root file is then used in the next step for the cocktail generation.

Cocktail generation (locally and on the grid)

The generation of the cocktail, of course, is based on the same macros/classes if run locally or on the grid. The local test is useful to generate a few hundred events and check if your parametrizations are parsed correctly. The local running is handled by runStandaloneCocktail.C which only requires minor modifications to work with new parametrizations. Its base arguments are the amount of events that should be generated (per event `nParticles = 250` (default) particles per species are generated flat in pT, rapidity and azimuth), the collision system (0: pp, 1: pPb, 2:PbPb) and the selection of particles to be generated. This selection, e.g. `262079` is the decimal number corresponding to the binary `111111111110111111` . Each one represents an activated particle for the simulation and examples can be found at the end of the macro.

Slight modifications need to be made where the parametrizations are loaded from the file created in the previous step.

```
gener = AddMCEMCocktailV2(200,0,decayMode,motherSelect,"../parametrizations/pp_5TeV.root","5TeV_PCMEMCal",nParticles,0.,50,10000,0,1,1,0);
```

The AliGenerator needs to be set to the right parametrization file, folder, number of particles and pT range (here 0-50 GeV/c). The GammaCocktail and HadronicCocktail tasks are already added and can be modified for different rapidity ranges if necessary.

The cocktail is then simply run by the following line to generate 50 events:

```
aliroot -x -l -b -q 'runStandaloneCocktail.C(50)'
```

On the grid, the procedure requires the parametrization file to be comitted to AliPhysics into the folder AliPhysics/PWG/Cocktail/parametrisations/ where one can either update the existing file or recreate the complete file. To have the cocktail produced just let the trainoperators (alice-pwg-GA-train-operators@cern.ch) know which existing or new dataset should be used. Effectively, the full information of the AliGenerator (gener) needs to be handed to the operator. Cocktails on the grid are generated with 1M events and show nearly no remaining statistical uncertainties. For a thorough analysis, the systematics of the cocktail have to be requested as soon as the main cocktail is validated.
It is advised to run in parallel the pure photon cocktail as well as the hadronic cocktail analysis tasks. When the data set has been activated ask the operators to activate the wagons: Pi0Cocktail_diffRap (& GammaCocktail_diffRap and run the corresponding train.

Addition: Inter/Extrapolation of particle spectra

This step is an important addition for analyses that require a reliable cocktail early on (before other particles are measured/published). For example in the 5 TeV pp case, there were no measured spectra of kaons, protons, phi, omega, ... available, but a first cocktail for the measurements was needed. There is the possibility to use mT scaling for all spectra but this has proven to yield very limited results at LHC energies (see https://arxiv.org/abs/1710.01933). Therefore it might be desired to inter/extrapolate the particle spectra from measurements at other center of mass energies. In the 5 TeV pp case, there are measurements available at 0.9, 2.76, 7 and 8 TeV which will allow for a very stable interpolation over a wide pT range. In order for the inter-/extrapolation-macro to work properly a global

binning for the desired particle (i.e. K^0_S, $K^{\pm}$, Λ, P, π^0, $\pi^{\pm}$, η, ...) needs to be defined for the corresponding energy. This should be done in the function `GetBinning()` of **CommonHeaders/ExtractSignalBinning.C** within the AnalysisSoftware repository.

The interpolations are handled by CalculateReference.C in the AnalysisSoftware repository. The macro is steered with an input text file and macro parameters. The text file is the most important ingredient as it provides the input spectra.

```
CombinedResultsPaperPP2760GeV_2017_04_19_FrediV2Clusterizer.root 0 Pi02.76TeV/graphInvCrossSectionPi0PCM2760GeVStatErr 0 Pi02.76TeV/graphInvCrossSectionPi0PCM2760GeVSysErr 2.76TeV 2760 1 bla bla SystematicsInputOtherEnergies/SystematicErrorAveragedSinglePCM_Pi0_2_76TeV_2016_12_14.dat ExternalInput/Theory/TheoryCompilationPP.root histoInvSecPythia8Monash2013LegoPi02760GeV
```

For each energy, a separate line in the above format has to be provided starting with the root file first, followed by a switch (0:histogram, 1:graph) and the name for the statistical error input (full path in root file including folder). Afterwards the same is given for the systematic errors. Then the center of mass energy in two forms is given (once in the standard format for our framework e.g. 900GeV, 7TeV, ... and once in units of GeV e.g. 900, 7000, ...). The following three arguments are usually set as "1 bla bla" followed by a systematic uncertainty input file (or also “bla”) and the corresponding theory curve.

The inter- or extrapolation can be made as soon as two or more energies are defined. It is run via:

```
root -b -x -q -l 'CalculateReference.C+\("configInterPolationPi0Comb.txt", 3,"Pi0"
,"pdf","5TeV",5023,20,kTRUE,"5TeV","","","",1000,"oHag"\)'
```

The arguments of CalculateReference.C are first the configuration file that contains the different energies, the number of defined energies in the file, the meson (here “Pi0”, but many others are already defined), the figure suffix, the energy string followed by the energy in GeV. The mode for the reference spectrum (e.g. 20 for a combined spectrum), a switch to use a special binning (should always be kTRUE), the energy from which that binning should be taken (see ExtractSignalBinning.h), three empty strings (not relevant for now), the number of trials (default=1000) and the fit function that should be used (here a modified hagedorn is used) are set after that.

Running the macro will now interpolate a spectrum at 5023 GeV from the input spectra defined in the text file. This is done by a binwise powerlaw fit. Figures are written to the folder pdf/2017_month_day/CalculateReference. After the first running, the fits need to be checked which can be done with for example input_Pi0EMCpp2760GeV_withFit_ratio.pdf which is created for each energy individually. If all fits looks good, then the figures for the binwise statistical and systematic inter/extrapolation can be checked (Pi0_EMC_Stat_Pt_vs_Sqrts.pdf and Pi0_EMC_Syst_Pt_vs_Sqrts.pdf). If problems at low or high pT are observed in these binwise plots, then the starting bin, the total amount of bins or the binning itself need to be adjusted inExtractSignalBinning.h.The remaining figures provide information about the resulting spectrum, its systematics and how it compares to the input spectra. The final result can be found in Interpolation.root.

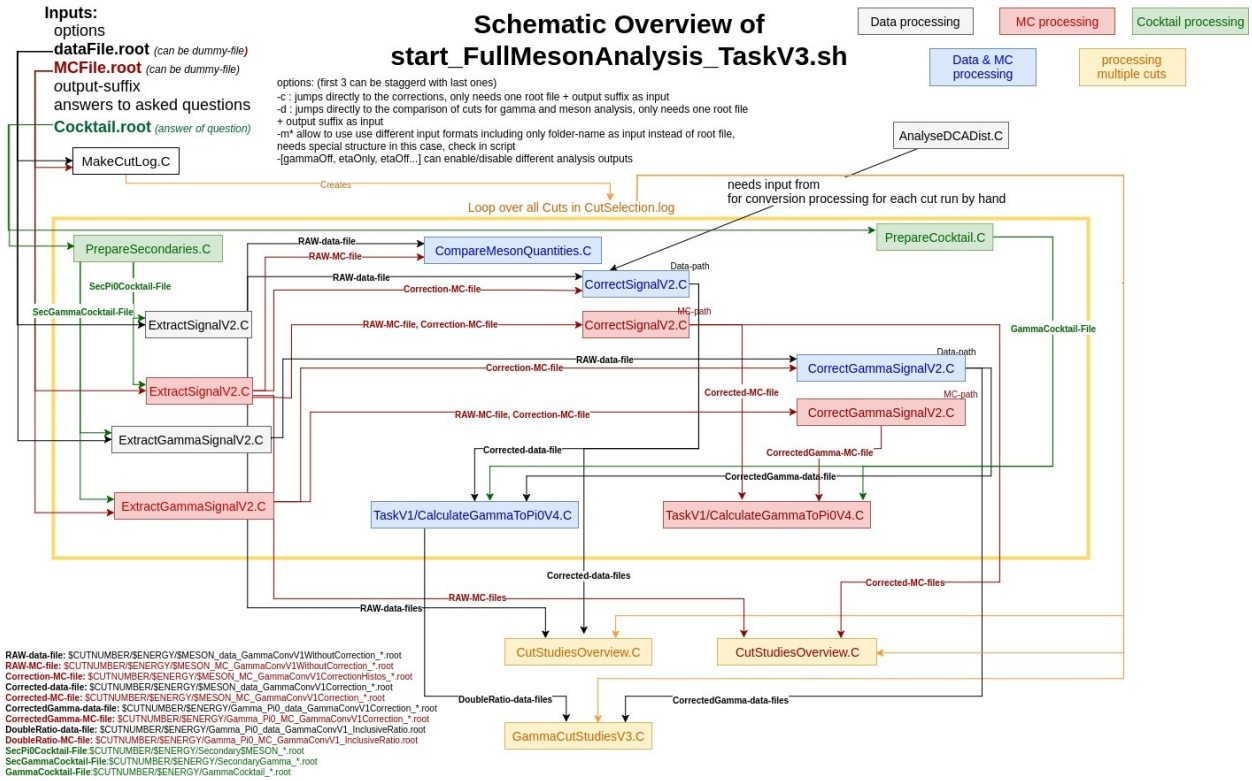
Before you start the analysis, get input .root files from grid and change their format by using the macros provide in the folder **AnalysisSoftware/DownloadAndDataPrep/** or **AnalysisSoftware/TaskQA**, as describe in the QA section. The neutral pion and eta analysis in the $\gamma\gamma$ channel is based on the analysis tasks:

- AliAnalysisTaskGammaConvV1.cxx
- AliAnalysisTaskGammaCalo.cxx
- AliAnalysisTaskGammaConvCalo.cxx

Each of them produces its own output files, which have to be provide to the *start_FullMesonAnalysis_TaskV3.sh*. The script can be started as follows:

```
start_FullMesonAnalysis_TaskV3.sh [-$OPTION] $Data-file.root [$MC-file.root] eps
```

If for some reason no data-file or MC-file is available you can give the script a dummy-file it will automatically detect this and try to avoid the corresponding macro running. It could however be that there are some errors in these cases and this option should only be used as a first trial not the general modus operandi. There are several specific options available to either jump to a specific step of the analysis (`-d,-c`) or switch on and off the gamma, pion or eta analysis (`etaOff,etaOnly,gammaOff,gammaOnly,pi0etaOnly,pi0Only`). The latter can be appended to he first two, but the option string has to start with a "-". If you want to jump to a more step further along the processing line you only have to hand it one root file and even that can be a dummy file if the *CutSelection.log* file contains the desired and valid Cutnumbers. In these case make sure the output files from the previous steps are contained in the current directory, where you are working. Otherwise the operation will fail. As already described most of the macros started by this shell-script are contained in the folder **AnalysisSoftware/TaskV1** and the general strategy for running is displayed in



It is recommended that you start the shell-script once per hand for each new analysis, which you do, as it will ask you questions, which you need to answer. These questions depend on the data set and collision system you have chosen, as well as the options you have given to the macro. For instance the questions asked for the pPb-direct photon analysis in the PCM-EMC analysis stream are (with the corresponding answers):

```
$ bash start_FullMesonAnalysis_TaskV3.sh -etaOff $referenceDirectoryData/GammaConvCalo_LHC13bc-pass2_20_A.root $referenc
eDirectoryMC/GammaConvCalo_MC_LHC13b2_efix_p1_p2_p3_p4_20_A.root eps
-etaOff
/mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-newDefaultPlusSys/GammaConvCalo_LHC13bc-pass2_20_A.r
oot
/mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-newDefaultPlusSys/GammaConvCalo_MC_LHC13b2_efix_p1_p
2_p3_p4_20_A.root
eps
eta calculation switched off
The data file specified is /mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-newDefaultPlusSys/GammaCo
nvCalo_LHC13bc-pass2_20_A.root
The MC file specified is /mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-newDefaultPlusSys/GammaConv
Calo_MC_LHC13b2_efix_p1_p2_p3_p4_20_A.root
Which mode are you running? 0 (PCM-PCM), 1 (PCM-Dalitz), 2 (PCM-EMCAL), 3 (PCM-PHOS), 4 (EMCAL-EMCAL), 5 (PHOS-PHOS), 9
(old files), 10 (EMC-merged), 11 (PHOS-merged), 12 (DCal-DCal), 13 (PCM-DCal)
2
You are analysing PCM-EMCAL output
Do you want to take an already exitsting CutSelection.log-file. Yes/No
N
Which collision system do you want to process? 13TeV (pp@13TeV), 13TeVLowB (pp@13TeV), 8TeV (pp@8TeV), 7TeV (pp@7TeV), 9
00GeV (pp@900GeV), 2.76TeV (pp@2.76TeV), 5TeV (pp@5.02TeV),  PbPb_2.76TeV (PbPb@2.76TeV), PbPb_5.02TeV (PbPb@5.02TeV), X
eXe_5.44TeV(XeXe@5.44TeV), pPb_5.023TeV (pPb@5.023TeV)
pPb5
The collision system has been selected to be pPb_5.023TeV.
Is a cocktail file available? Yes/No?
Y
Please enter the filepath of the cocktail file.
/home/fbock/Photon/Software/PCGGIT/CocktailGridRuns/EMCocktail_pPb5TeV_PCMEMC_defaultParam_20170901.root
The cocktail file specified is /home/fbock/Photon/Software/PCGGIT/CocktailGridRuns/EMCocktail_pPb5TeV_PCMEMC_defaultPara
m_20170901.root
Please enter the rapidity used in the cocktail, e.g. 0.80
0.80
Rapidity of 0.80 has been chosen.
Do you want to produce Direct Photon plots? Yes/No?
YPCMEMC
Will produce Direct Photon plots with special PCMEMC binning...
How many p_T bins do you want to use for Pi0/direct photon?  21 (8 GeV), 22 (10 GeV), 23 (14 GeV), PCM-EMC 33
28
You have chosen  28  pt bins for pi0 in direct photon binning
Do you want to use MinBias Efficiencies only? Yes/No?
Y
Calculating MinBias Efficiecy only ...
Do you want to use THnSparse for the background? Yes/No?
N
Will NOT use THnSparse for the background ...
mode has been chosen: 2
I went into standard modes
Which fit do you want to do? CrystalBall or gaussian convoluted with an exponential function? CrystalBall/Gaussian?
G
Gaussian chosen ...
Please check that you really want to process all cuts, otherwise change the CutSelection.log. Remember at first all gamm
a cutstudies will be carried out. Make sure that the standard cut is the first in the file. Continue? Yes/No?
Y
```

After you have understood and verified that the answers you gave, have the desired outcome you can also store them in a text file ($ENTER-separated) like answersExamplepPbPCMEMC.txt and start the script like this:

```
 bash start_FullMesonAnalysis_TaskV3.sh -etaOff $referenceDirectoryData/GammaConvCalo_LHC13bc-pass2_20_A.root $reference
DirectoryMC/GammaConvCalo_MC_LHC13b2_efix_p1_p2_p3_p4_20_A.root eps < answersExamplepPbPCMEMC.txt
```

This option makes it possible to run many different start scripts after one another started by a common shell script without needing to interact with it. In general it is recommended to have one additional bash script, where you keep track what you usually use as input files and which analysis you ran in a certain directory (i.e. runBaseCuts.sh). This also helps to keep track of your files and reduces the probability of creating an error during the configuration of the

shell script.

In the following we will go through the different basic macros started by the shell-script and show some characteristic plots produced by them. In general it is recommended to have a look at all plots, log-files and couts which are produced by the various macros in particular for new-commers, as these are there for a reason and should not be ignored. We strongly recommened, however, not to reinvent the wheel and simply use the macros and ask if there is an output which you are missing or which is not clear to you. This will help us improve the software and you to save time (not wasting your energy on redesigning the plotting routines, as the later are in most cases already pretty good and have been used for the publications of the PWGGA).
If you find a bug in the software, please let us know as it might not only affect you but also others and as such can have quite a large impact. Remember to also take this into account if you make changes, we all benefit from a common framework but this means we all need to do our best to keep it running and improve it. **YOU ARE NOT JUST A USER, YOU ARE A DEVELOPER!** So please make sure your changes don't interfer with someone else working code. Everyone else will try to do the same for you!

A nice little 'helper-macro' is the *MakeCutLog.C*, which tells you which cuts are contained in the corresponding file. It can be started with:

```
root -b -x -q -l 'TaskV1/MakeCutLog.C("file.root","CutSelection.log",$mode)'
```

It will write a file called *CutSelection.log*, as defined by the argument. It is called by default when executing our base shell-script and call cuts contained in this cutselection file will be analyzed. If you don't want that, you explicitly have to tell this when asked about this particular file and edit it yourself.

The modes which are defined in the conversion software are:

mode	reco technique
0	PCM-PCM
1	PCM-Dalitz
2	PCM-EMC
3	PCM-PHOS
4	EMC-EMC
5	PHOS-PHOS
6	EMC-Dalitz
7	PHOS-Dalitz
10	mEMC
11	mPHOS
12	DMC-DMC
13	PCM-DMC

Neutral Pion and Eta Meson Signal Extraction

ExtractSignalV2.C

This macro is supposed to:

- analyse the invariant mass of the mesons for each pT bin
- subtract the mixed event BG after proper normalization,
- fit the peak with different peak functions and
- extract the raw yields

in case a data file is processed. If it is a MC file it does all this and extracts the correction factors (acceptance, efficiency, secondary contamination,...) in addition. Furthermore, different estimates of the additional BG subtraction are computed and the signal is extracted in different integration windows around the fitted meson peak to later determine the signal extraction uncertainty.

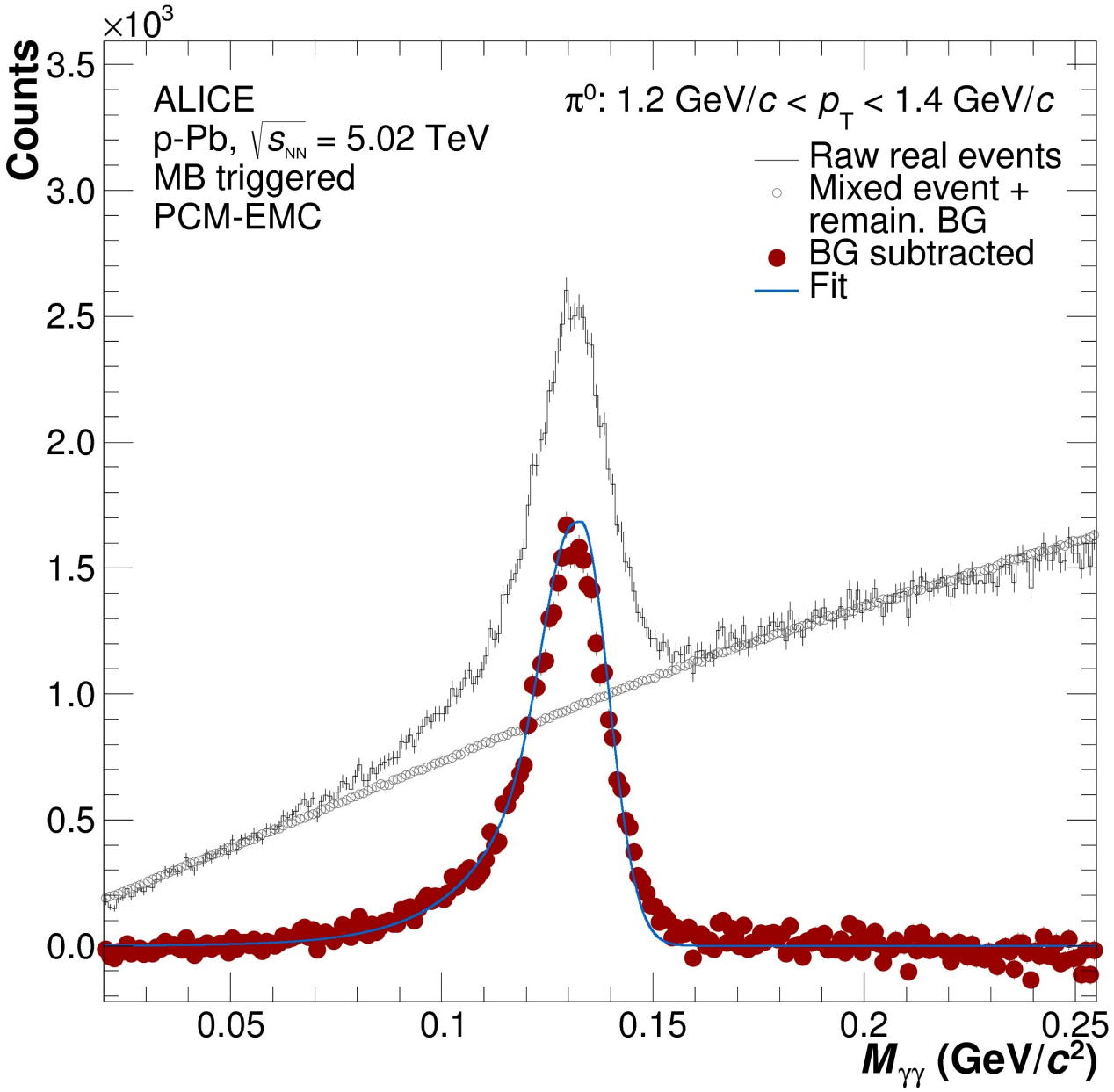
The p_T-binning for the π^0/η is defined in the header *ExtractSignalBinning.h*. The macro can be started on a standalone basis as follows:

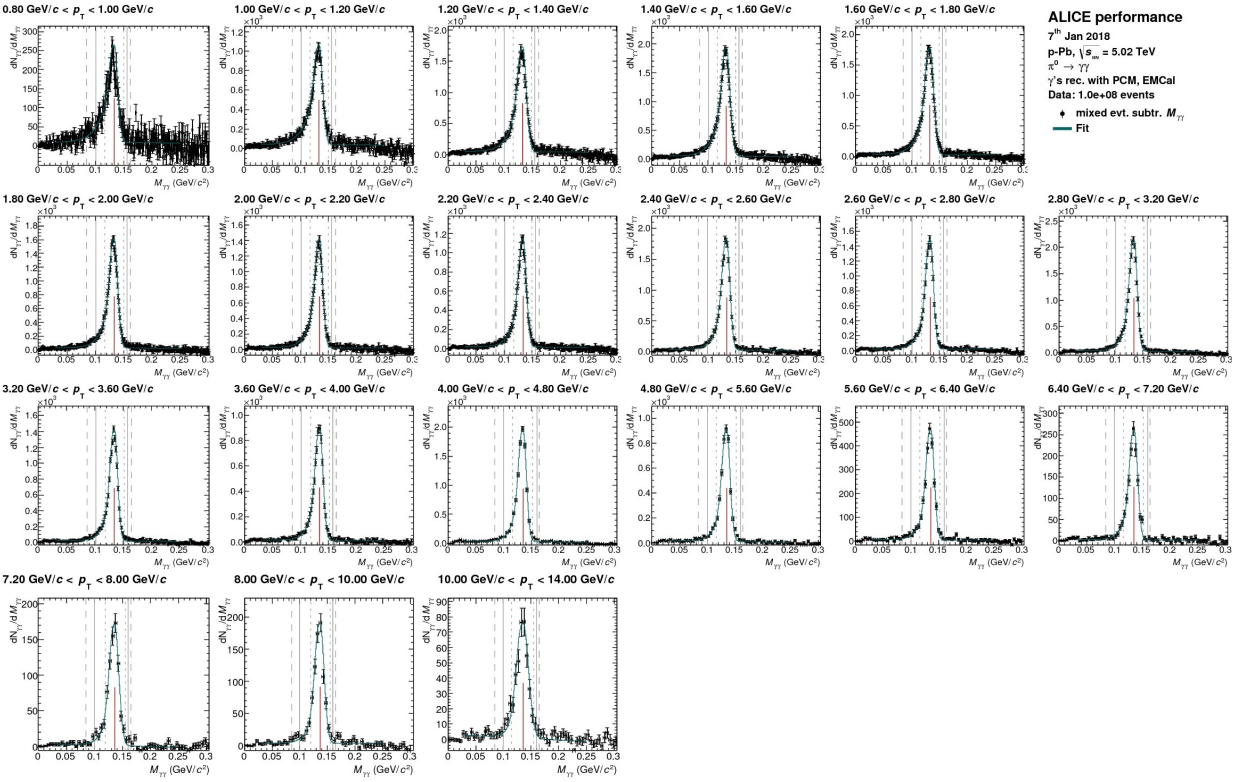
```
root -b -x- q -l 'TaskV1/ExtractSignalV2.C+("$MESONAME","#ROOT-File","$CUTNUMBER","$SUFFIX","$MCOPTION","$ENERGY","Gauss
ian","$OPTIONSA","$OPTIONSB","","",$NPTBINS,$OPTIONADDSIG,$MODE,0,$OPTIONTHNSPARSE)'
```

for the data set above the data and MC would have been called as follows:

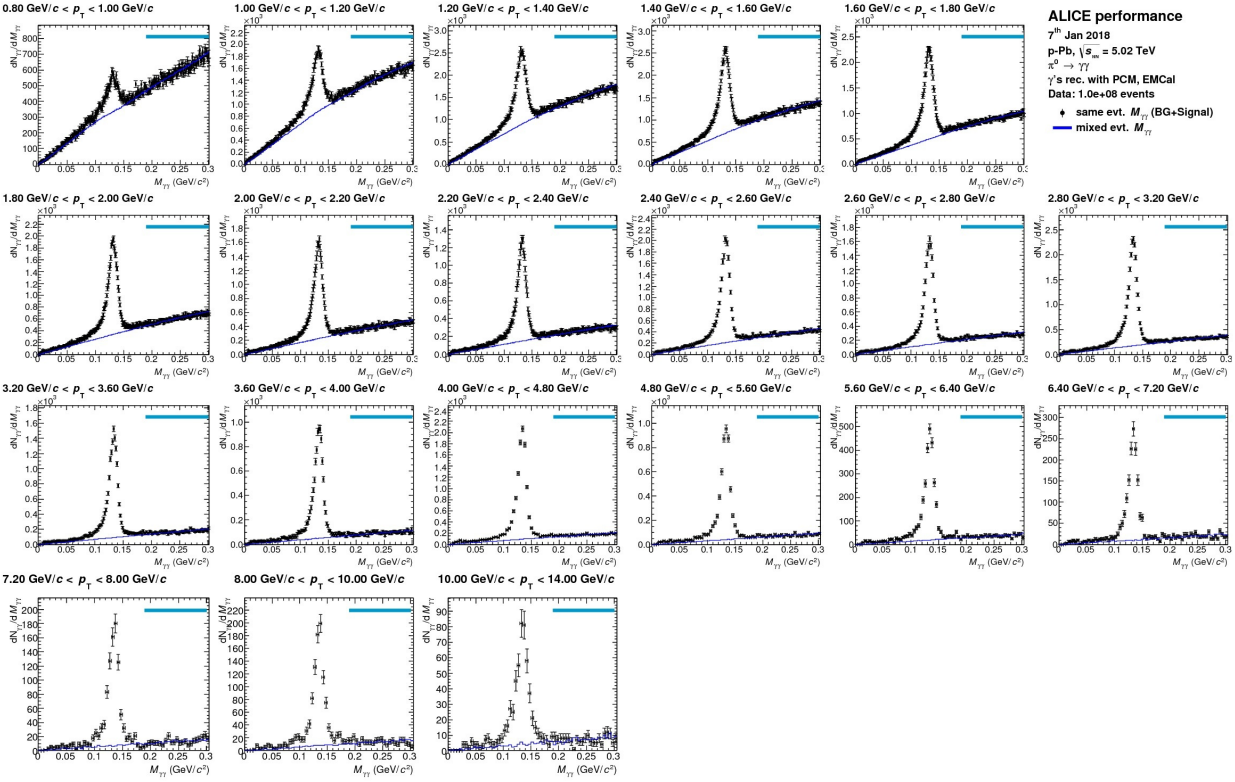
```
root -b -x- q -l 'TaskV1/ExtractSignalV2.C+("Pi0","/mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-n
ewDefaultPlusSys/GammaConvCalo_LHC13bc-pass2_20_A.root","80000113_00200009327000008250400000_1111141057032230000_0163103
100000010","eps","kFALSE","pPb_5.023TeV","Gaussian","directPhotonA","MinBiasEffOnly","","",28,kFALSE,2,0)'
root -b -x- q -l 'TaskV1/ExtractSignalV2.C+("Pi0","/mnt/additionalStorage/OutputLegoTrains/pPb/Legotrain-vAN20170525FF-n
ewDefaultPlusSys/GammaConvCalo_MC_LHC13b2_efix_p1_p2_p3_p4_20_A.root","80000113_00200009327000008250400000_1111141057032
230000_0163103100000010","eps","kTRUE","pPb_5.023TeV","Gaussian","directPhotonA","MinBiasEffOnly","","",28,kFALSE,2,0)'
```

For the details on the options you can have a look at the code. For beginners, it is however highly recommended to start it once with the shell-script to obtain the proper options. The macro will write its plots to **$CUTNUMBER/$ENERGY/$SUFFIX/ExtractSignal** and the corresponding subfolders. The most important output will look like the following plots and all of them should be studied in detail for all cuts and mesons. Also some trending plots are stored in the the corresponding subfolder. These will allow you to judge whether the parameter-settings were reasonable and the fits didn't run into any particular limits.

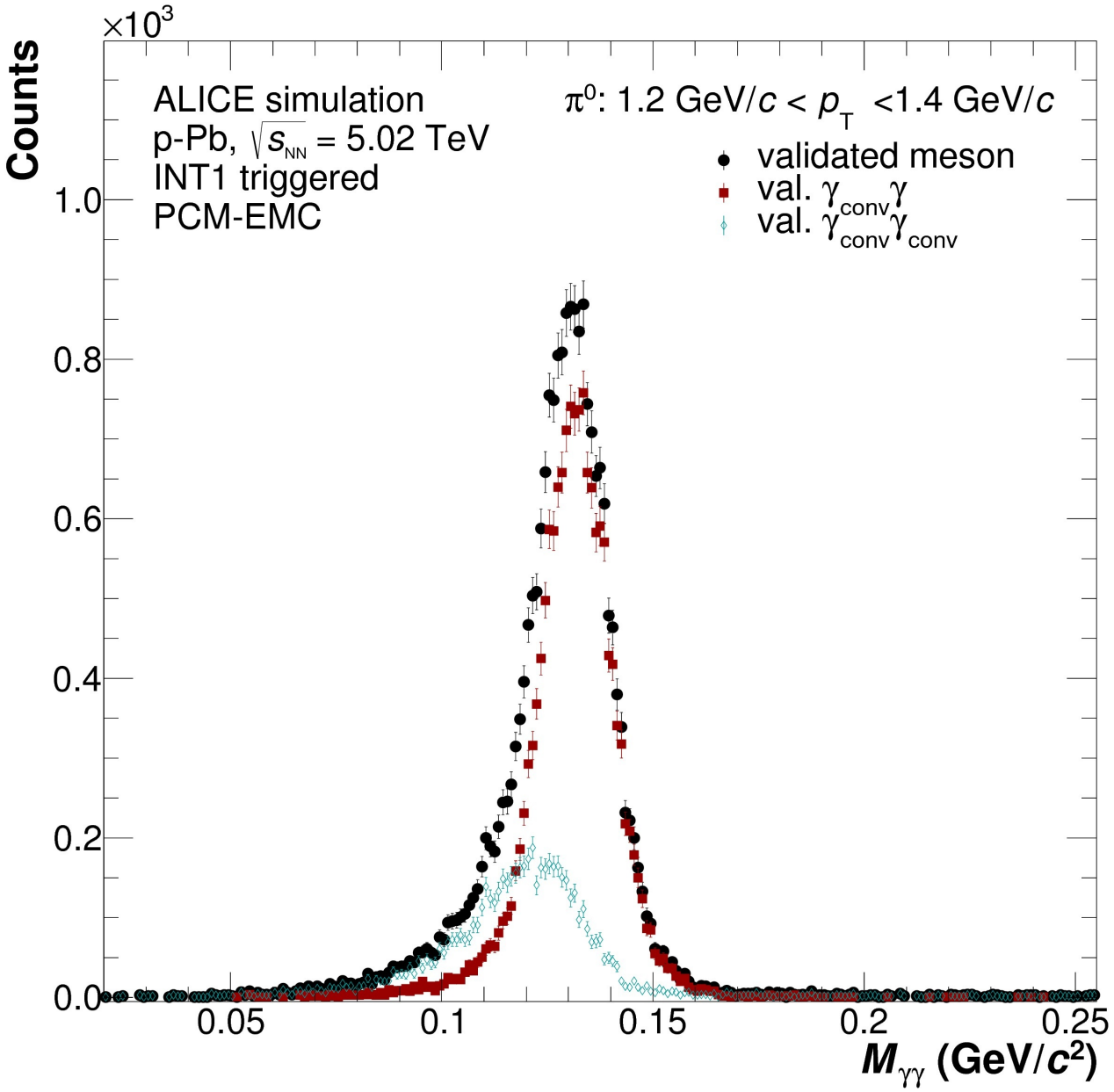

Counts
×10^3
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
ALICE
p-Pb, √s_NN = 5.02 TeV
MB triggered
PCM-EMC
π^0: 1.2 GeV/c < p_T < 1.4 GeV/c
Raw real events
Mixed event + remain. BG
BG subtracted
Fit
0.05
0.1
0.15
0.2
0.25
M_γγ (GeV/c^2)

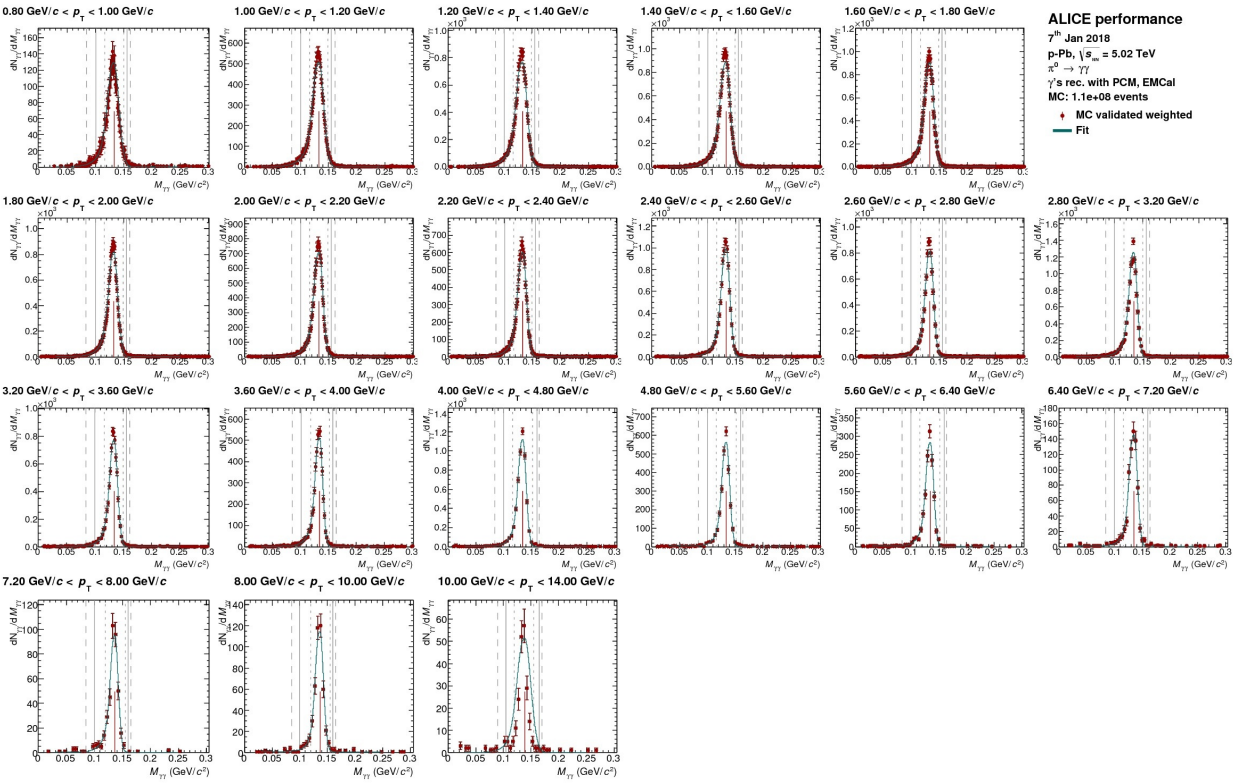


ALICE performance
7th Jan 2018
p-Pb, $\sqrt{s_{NN}}$ = 5.02 TeV
$\pi^0 \rightarrow \gamma\gamma$
γ's rec. with PCM, EMCal
Data: 1.0e+08 events
mixed evt. subtr. $M_{\gamma\gamma}$
Fit



ALICE performance
7th Jan 2018
p-Pb, $\sqrt{s_{NN}}$ = 5.02 TeV
$\pi^0 \rightarrow \gamma\gamma$
γ's rec. with PCM, EMCal
Data: 1.0e+08 events
same evt. $M_{\gamma\gamma}$ (BG+Signal)
mixed evt. $M_{\gamma\gamma}$


Counts
×10^3
ALICE simulation
p-Pb, √s_NN = 5.02 TeV
INT1 triggered
PCM-EMC
π^0: 1.2 GeV/c < p_T <1.4 GeV/c
validated meson
val. γ_conv γ
val. γ_conv γ_conv
1.0
0.8
0.6
0.4
0.2
0.0
0.05
0.1
0.15
0.2
0.25
M_γγ (GeV/c^2)



In addition it writes its log-files as well as the root-files to **$CUTNUMBER/$ENERGY/** they are called:

```
$MESONNAME_[data,MC]_EffiCheck_RAWDATA_$CUTNUMBER.dat
$MESONNAME_[data,MC]_FileErrLog_80000113_$CUTNUMBER.dat
$MESONNAME_[data,MC]_GammaConvV1WithoutCorrection_$CUTNUMBER.root
$MESONNAME_[data,MC]_GammaExtractionEffiCheck_RAWDATA_$CUTNUMBER.dat
$MESONNAME_MC_GammaConvV1CorrectionHistos_$CUTNUMBER.root
```

!!! Very important: !!!

Check the $CUTNUMBERS/$ENERGY/$SUFFIX/EXTRACTSIGNAL/MONITORING folder to see if the mass peaks are fitted with high quality fits meaning:

- MESON Amplitude converged well for each bin
- MESON LambdaTail converged within its limits (this parameter can be fixed for higher pt if it is needed)
- MESON Sigma converged also within its limits and has a smooth behaviour as function of pt
- MESON Mass converged within limits and has a smooth behaviour as function of pt

Also check all the plots in $CUTNUMBERS/$ENERGY/$SUFFIX/EXTRACTSIGNAL/ to check if all the bins have sufficient statistics for both data and MC. Increase the pt bin widths if there is a lack statistics. The number of bins in the invariant mass plots can also be adjusted, make sure there are sufficients points to be fitted in the main signal peak.

AnalyseDCADist.C

This macro is especially need for the PCM π^0 and η analyses. In case of the PCM photons the association to the event is not as easy as for the EMCal photons as there is no direct variable which relates a certain PCM photon in terms of time to corresponding event. Especially photons which are reconstructed only based on the TPC tracks can orginate with a certain probabiltity from one of the neighboring bunch crossings. As a rule of thumb one can say that a TPC track coming from a collision 500ns after the triggering one will be displaced by about 1.5 cm in Z in the TPC compared to the ones from the triggering collisions. This is due to the drift time of the electrons of 2.7 cm/μs and the

total drift length of about 92 μs. The latter is approximately as long as 1 bunch needs to go round the LHC once. This mean that many tracks from neighboring bunches are actually contained in the TPC and as such also many photons. These tracks can only be rejected as pileup if they are not pointing to the primary vertex within a certain window, which needs to be larger for secondary tracks compared to primary tracks and depends on the primary vertex resoluton for the TPC-only tracks. To evaluate the contribution from these out-of-bunch photons to the neutral meson spectra the *AnalyseDCADist.C*-macro has been developed. Here we use the fact that a photon from another bunch-crossing will most likely have a different distribution in the distance of closest approach from the vertex in its

Z-coordinate (dca_Z). This contribution will depend on the photon quality (1: TPC only photon, 2: 1 leg rec with TPC only, 3: both legs based on partial ITS+TPC information) which enters for the neutral meson reconstruction. As such

6 different categories have to be considered for the π^0&η analysis. Within this macro the projections out of the dca-trees contained in the GammaConvV1_*.root for the dca_Z of photon candidates in the invariant mass region of the corresponding meson are filled for the different reconstruction qualities and then evaluated using a root internal 'background finding' routine (`TH1::ShowBackground()`). The macro can be executed as follows:

```
    root -b -x -q -l 'TaskV1/AnalyseDCADist.C+("$MESON","Data-main-file.root","Data-add-file.root","$CUTNUMBER","eps","$
ENERGY","",kFALSE,$NPTBINS,0)
```

It has to be run by hand as it most likely will need special adjustments for every energy and maybe even data-taking period. However, it is not necessary to run it for every possible cutvariaion when calculating the systematic uncertainties due to other sources. In this case you can simply copy the output-file of the standard cut ($CUTNUMBER/$ENERGY/$MESON/home/$MESON_Data_GammaConvV1DCATestAnalysed.root) into the

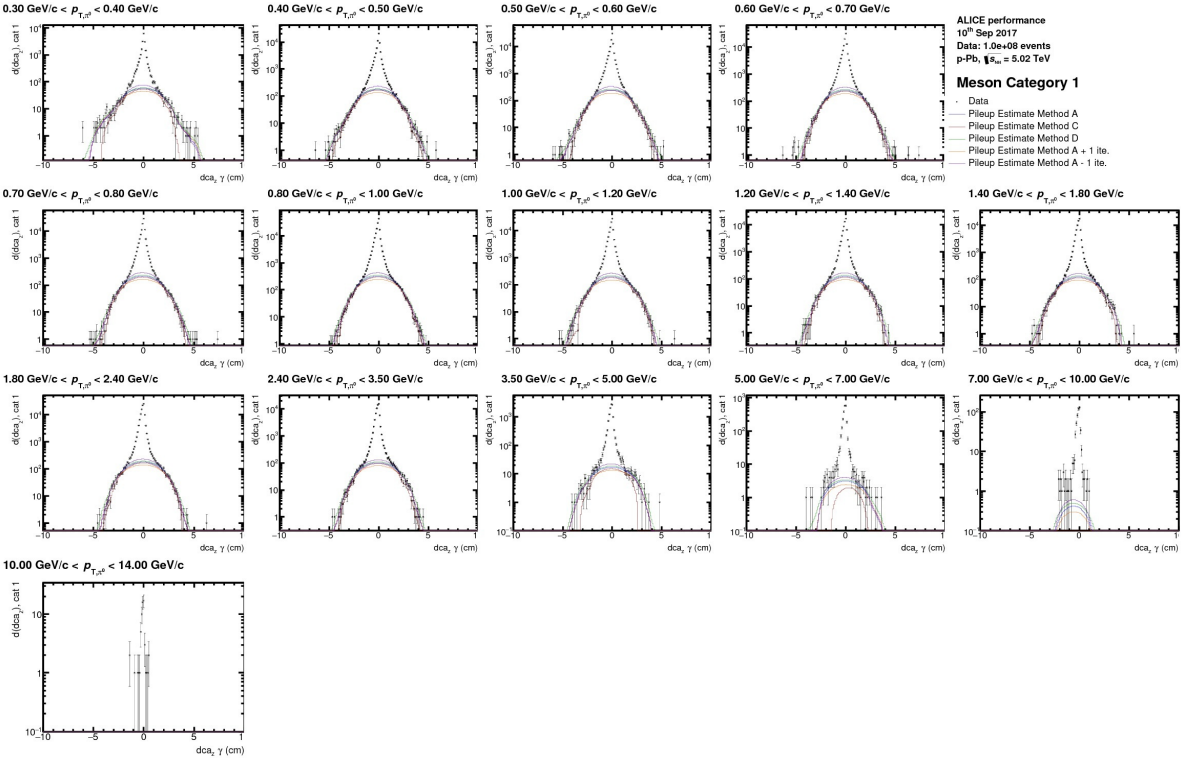
directories of the variations ($CUTNUMBERs/$ENERGY/). It has to be carefully checked that the 'wings' in all p_T bins

at higher $|dca_Z|$ or neither underestimated nor overestimated which might need some tweaking of the following parameters in CommonHeaders/ExtractSignalBinning.h:

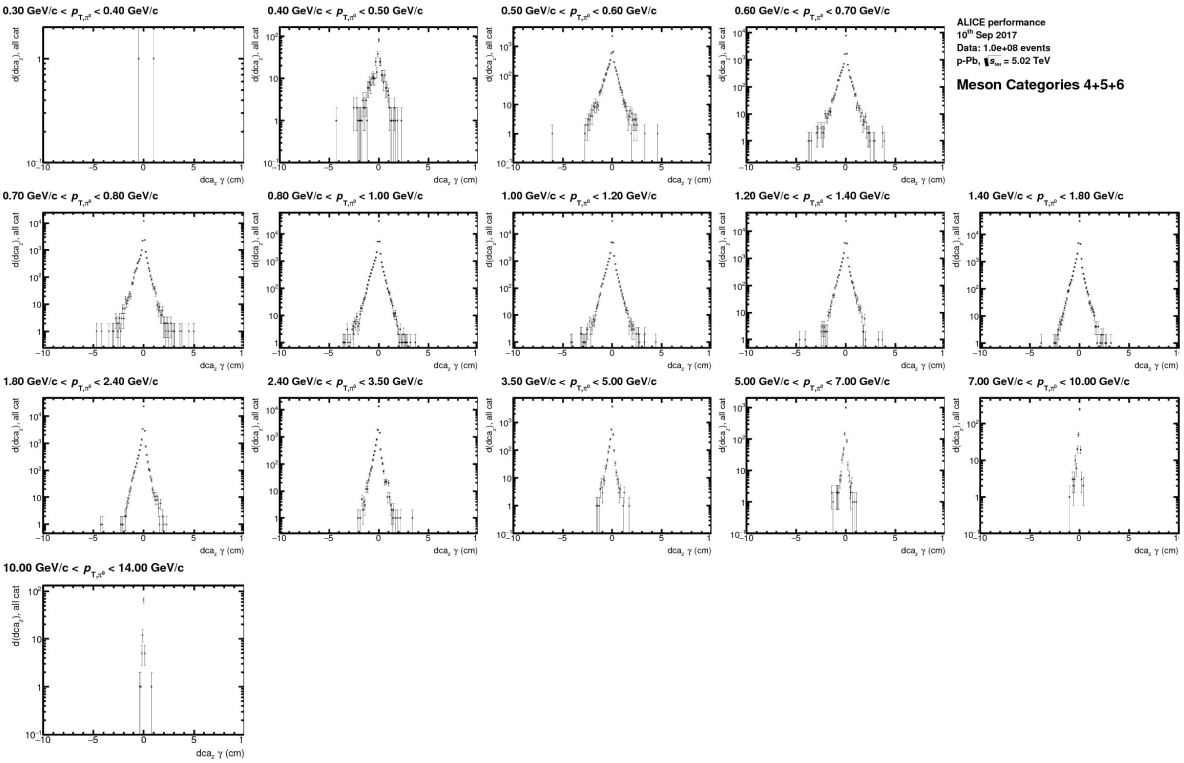
```
    fMaxYFracBGOverIntHist              = 50;
    nIterBGFit                          = 13;
    TString optionBGSmoothingStandard   = "BackSmoothing9";
    TString optionBGSmoothingVar1       = "BackSmoothing7";
    TString optionBGSmoothingVar2       = "BackSmoothing11";
```

Examples of optimized fits and the corresponding final correction factors can be found in the following plots or in the latest PCM analysis notes on pp 8Tev, pPb 5TeV or Pb-Pb 2.76TeV.

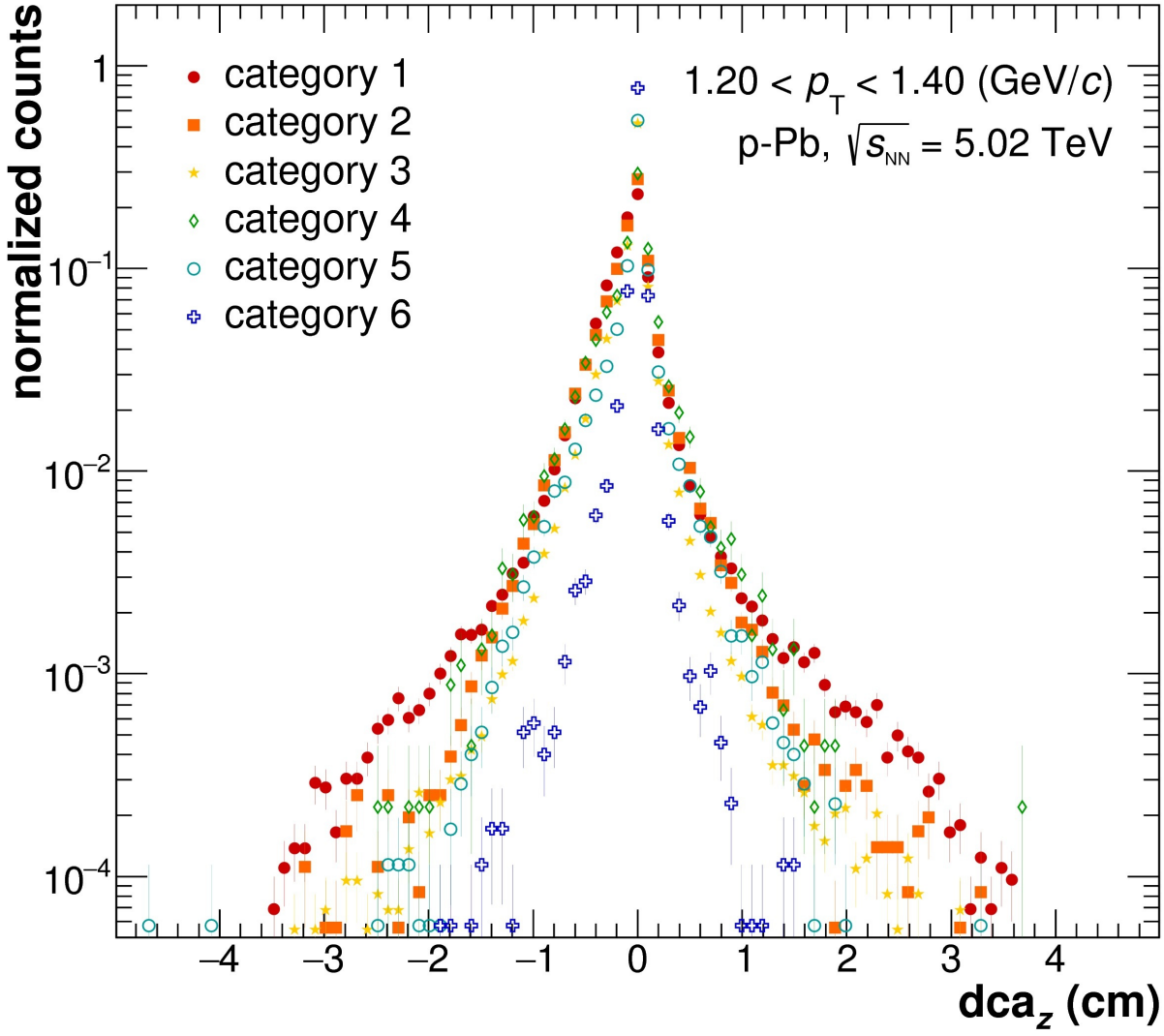
- worst DCA meson category (2* category 3 photons)



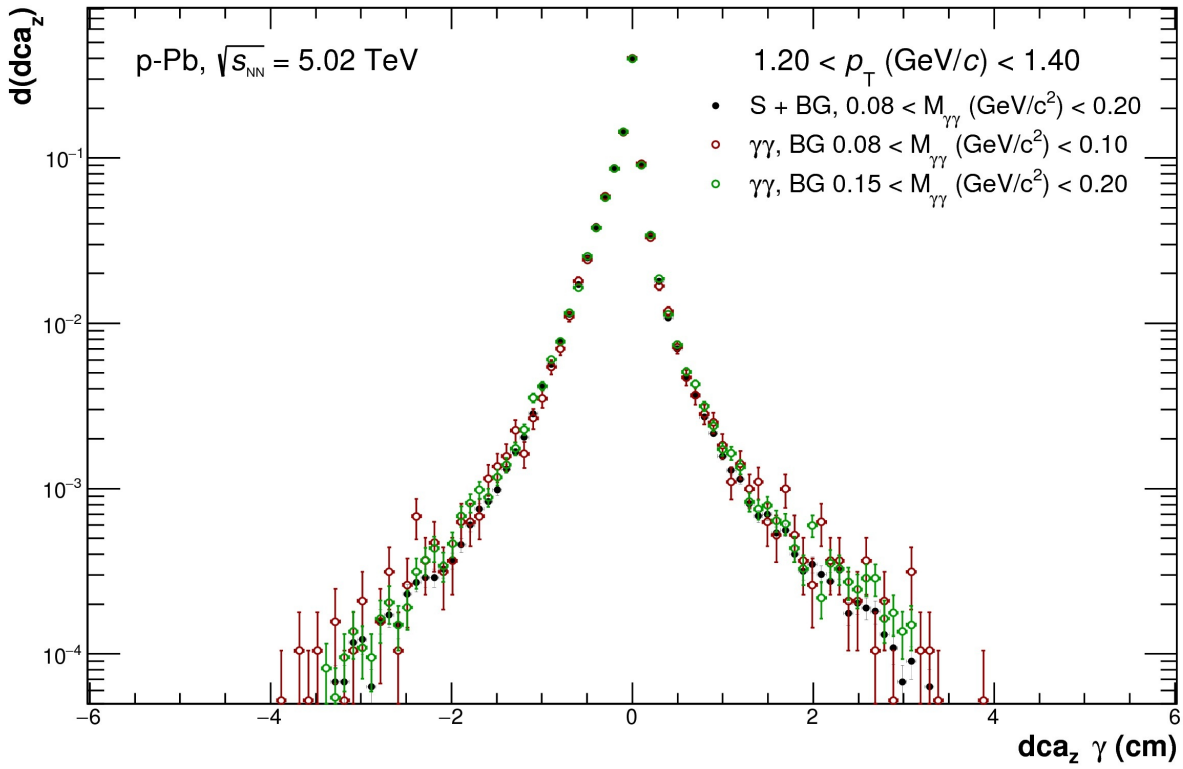
- best DCA meson categories (at least 1 photon with ITS info)



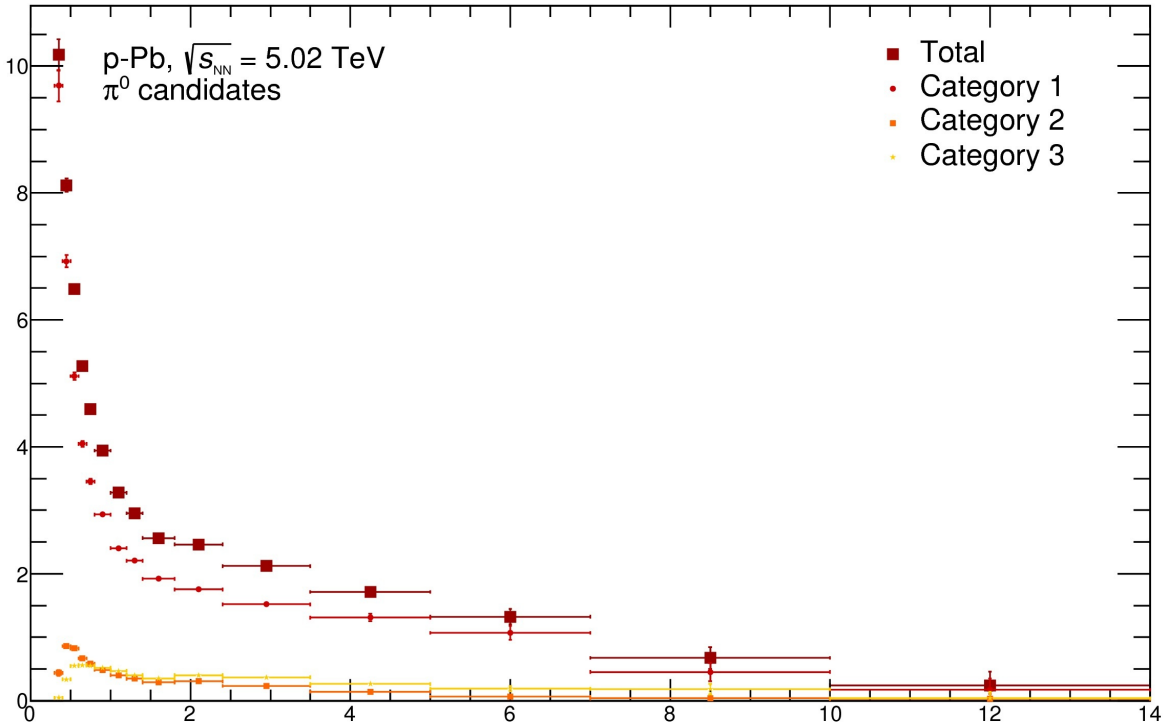
- all categories overlayed



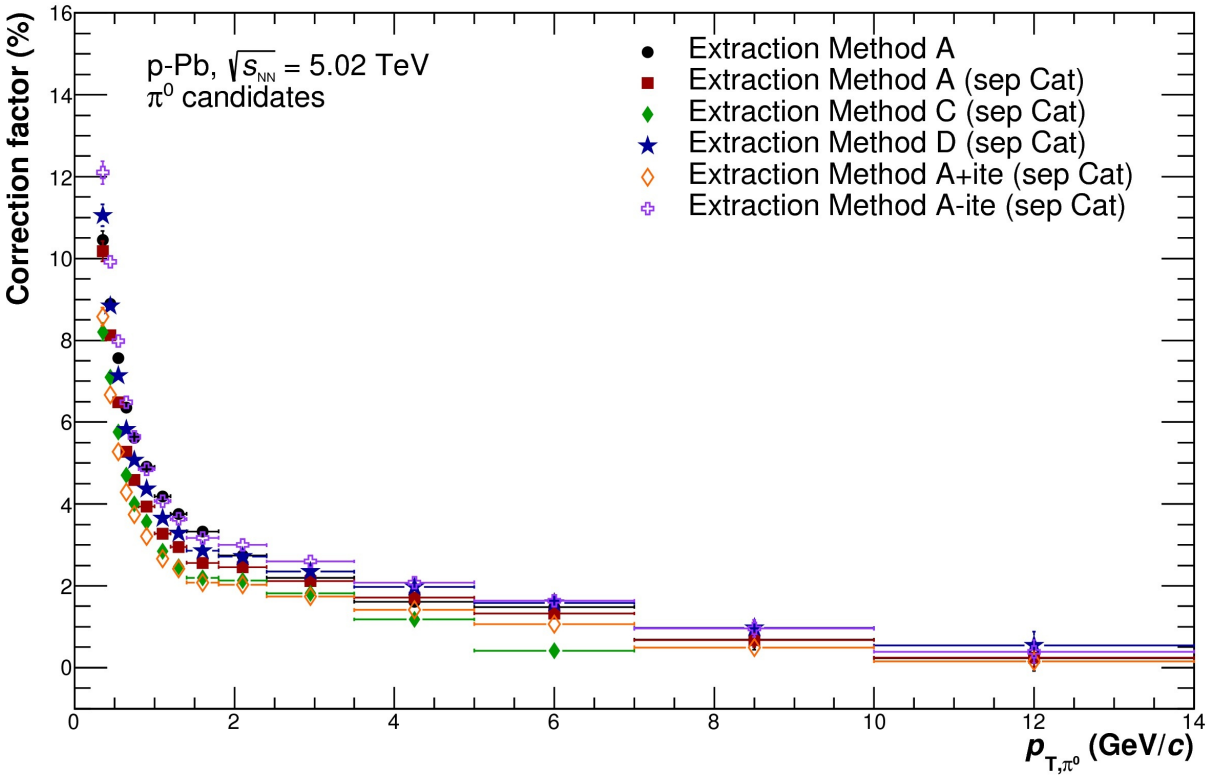
- $dcaz$ distribution in different invariant mass windows



- Calculated out-of-bunch pileup contribution for different categories separately



- correctly weighted out-of-bunch pielup contribution using different estimates



Meson Corrections

CorrectSignalV2.C

Applies the corrections to the raw yields; the input for this macro is the output from the *ExtractSignalV2.C* and

AnalyseDCATestV1.C. Also in this a .root file and plots are produced in the folder of the respective cutnumber.

The arguments of the function are

```
void  CorrectSignalV2(  TString fileNameUnCorrectedFile = "myOutput",
                        TString fileNameCorrectionFile  = "",
                        TString fCutSelection           = "",
                        TString suffix                  = "gif",
                        TString nameMeson               = "",
                        TString isMC                    = "",
                        TString optionEnergy            = "",
                        TString optionPeriod            = "",
                        TString fEstimatePileup         = "",
                        Bool_t optDalitz                = kFALSE,
                        Int_t mode                      = 9,
                        Int_t triggerSet                = -1

                    ){
```

The main script executes CorrectSignalV2.C automatically but it can be called independently with:

```
root -b -x- q -l 'TaskV1/CorrectSignalV2.C+("$FILETOCORRECT","$FILEWITHCORRECTIONS","$CUTNUMBER","$SUFFIX","$MESON","$IS
MC","$ENERGY","","",$DALITZ,$MODE)'
```

As an example:

```
root -b -x- q -l 'TaskV1/CorrectSignalV2.C+("80000113_11111410570a2230000_01631031000000d0/pPb_5.023TeV/Pi0_data_GammaCo
nvV1WithoutCorrection_80000113_11111410570a2230000_01631031000000d0.root","80000113_11111410570a2230000_01631031000000d0
/pPb_5.023TeV/Pi0_MC_GammaConvV1CorrectionHistos_80000113_11111410570a2230000_01631031000000d0.root","80000113_111114105
70a2230000_01631031000000d0","eps","Pi0","kFALSE","pPb_5.023TeV","","",kFALSE,4)'
```

Important output plots are (non exhaustive list):

- $MESON_Acceptance
- $MESON_TrueEffSimple
- $MESON_Mass
- $MESON_RatioMass
- $MESON_FWHM
- $MESON_RatioFWHM
- $MESON_Data_RawYieldDiffIntRanges
- $MESON_Data_CorrectedYield (Normal or True efficiency, always use True for PCM analyses and Normal for Calorimetric)

Questions to ask yourself while inspecting the histograms:

- Is the acceptance behaving like you expect
- Is the efficiency bahaving like it should; and is the difference between the Reconstructed and True efficiency reasonable?
- Is the reconstructed pi0 mass in data and MC close? Judge this also from the ratio plot. 1% offset means roughly 4% error in the corrected yield!!
- Is the FWHM simulated well in MC and does it behave as expected?
- Is the RawYield with different integration ranges stable?
- The corrected yield is only to be checked as last plot, it has no value unless everything else is of high quality!

Secondary Corrections for the neutral pion reconstruction

How To obtain the necessary Input Spectra & Cocktail Output

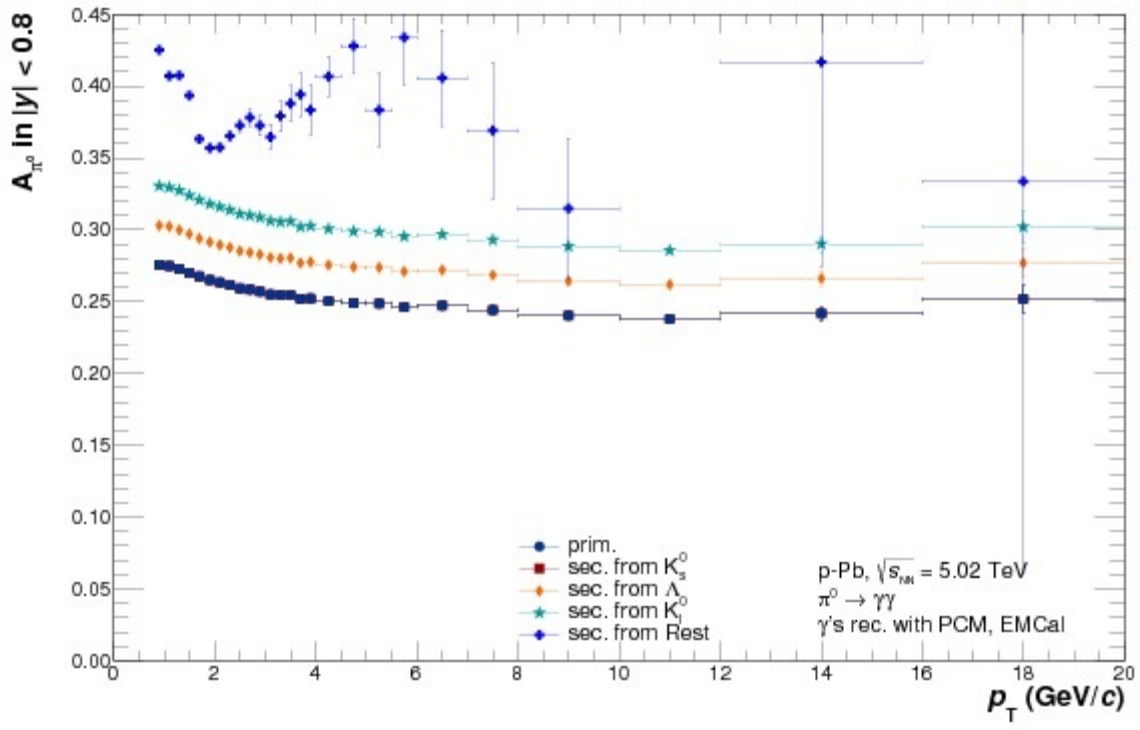
For the secondary correction to the π^0's and γ's we do not rely any longer purely on the simulations, except for the material interactions contribution, consequently additional experimental inputs are needed. For each collision system the measured spectra for K_s^0 & Λ will be needed (ask LF convenors for corresponding input if not already included in the cocktail framework). If these are not available in a certain collision system, energy or centrality one can either use the charged kaons as proxy for the K_s^0 or in the pp case inter/-extrapolate the kaons/Lambda's from other collision energies. For the later ideally all available collision energies are to be used (i.e 0.9, 2.76, 7, 13 TeV to get to 5 or 8 TeV). For the Lambda no other obvious candidate exists which could stand in as proxy for the Lambda if inter-/extrapolation is not possible. In that case one can use m_T scaling from the proton, this is however much less precise.

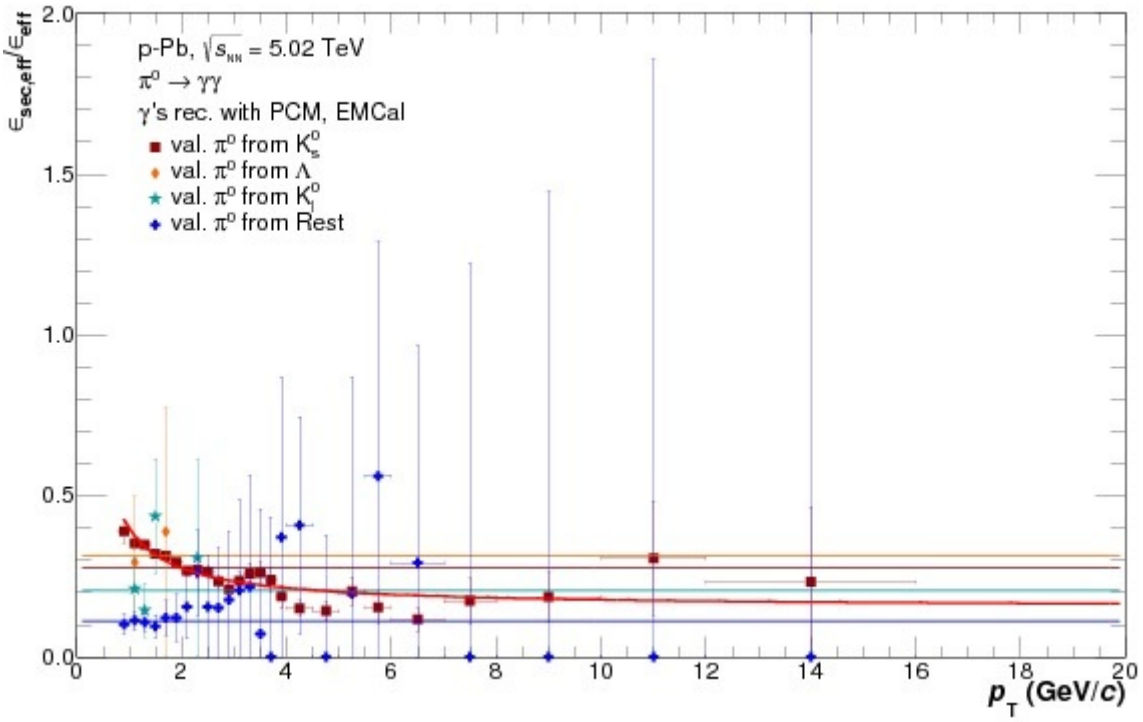
For the description of the interpolation procedure you can jump to the Cocktail Framework Intro. If m_T scaling is used there is no need to do calculate the spectrum separately it would be automatically if the Λ is enabled in the cocktail parametrization in cocktail-framework. A detailed description how the corresponding spectra need to be included in the cocktail frame work and how they are paremetrized is given in the section Cocktail Framework Intro along with the instructions on how to run the corresponding cocktail on the grid. However, you have to ensure the train-operator enables the wagon for the hadronic cocktail as well (Pi0Cocktail_diffRap), which will run several wagons of AliAnalysisTaskHadronicCocktailMC.cxx in parallel on the output of the generated EM-cocktail.
A general remark: Make sure your parametrization for all particles describe the spectra better than 5-10% over the full measured transverse momentum region, as otherwise the generated cocktail will certainly not reproduce the data. At low include (0,0) in the fitting and at high make sure the n of all spectra is approximately the same and they don't cross in the unmeasured region if they are not expected to.

Integration of the Cocktail-output in the Analysis-Flow for the Pi0's

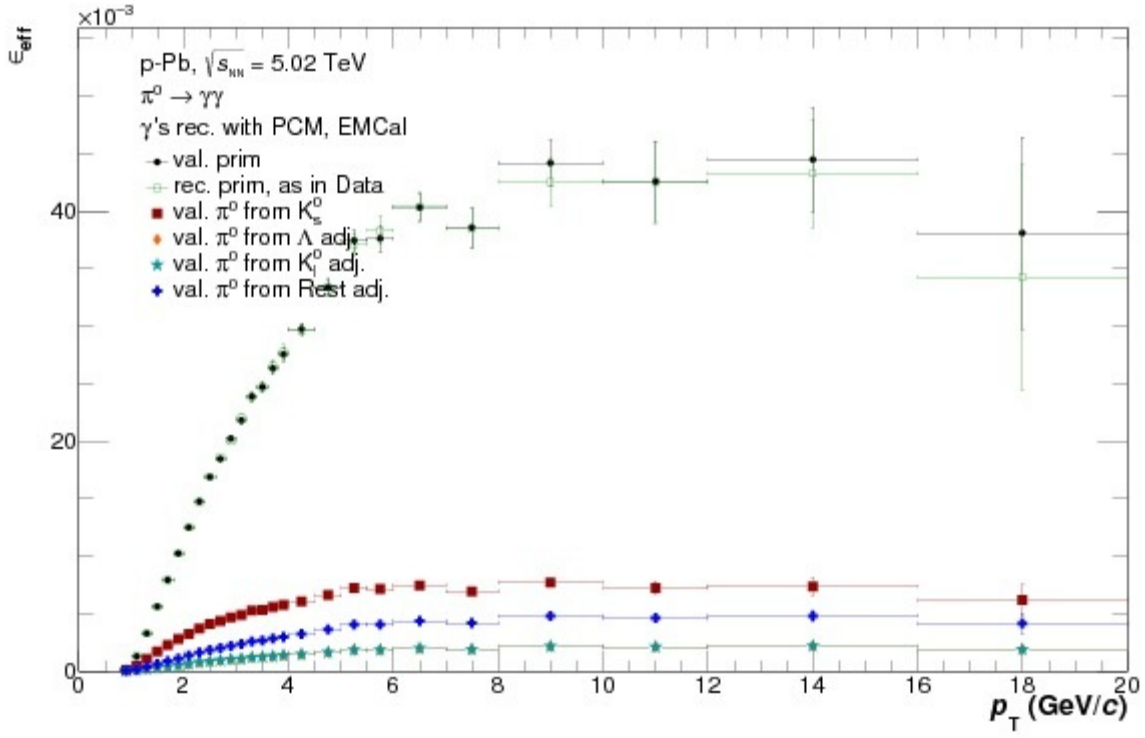
- Once the train finished or you generated the file on your computer, copy the *AnalysisResults.root* file to your computer and rename it ideally with the data of generation, system and settings short hand for bookeeping.
- When asked by the *start_FullMesonAnalysis_TaskV2.sh* whether you have a cocktail file answer "Yes" and answer the consecutive questions regarding the file location and rapidity settings. For the later the answer is "0.8" for most analysis and it depends on the rapidity range you are using in the corresponding cutselection and analysis.
- This should take care of nearly everything else necessary afterwards. However you have to control whether the secondary efficiency and acceptance make sense as these might need to be fixed due to lack of statistics in the corresponding MC. Have a look at the corresponding plots for that. Examples for the pion reco are shown below.

Pi0_AcceptanceWithSec_* & Pi0_RatioSecEffiToTrueEff_*

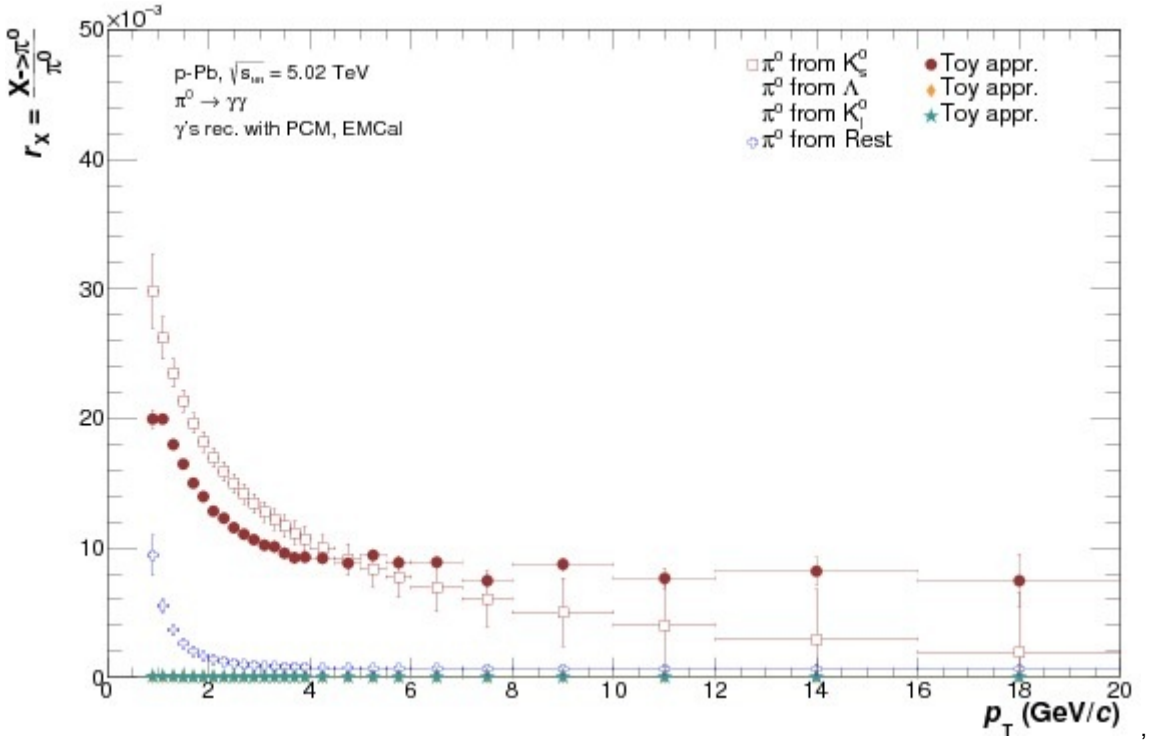




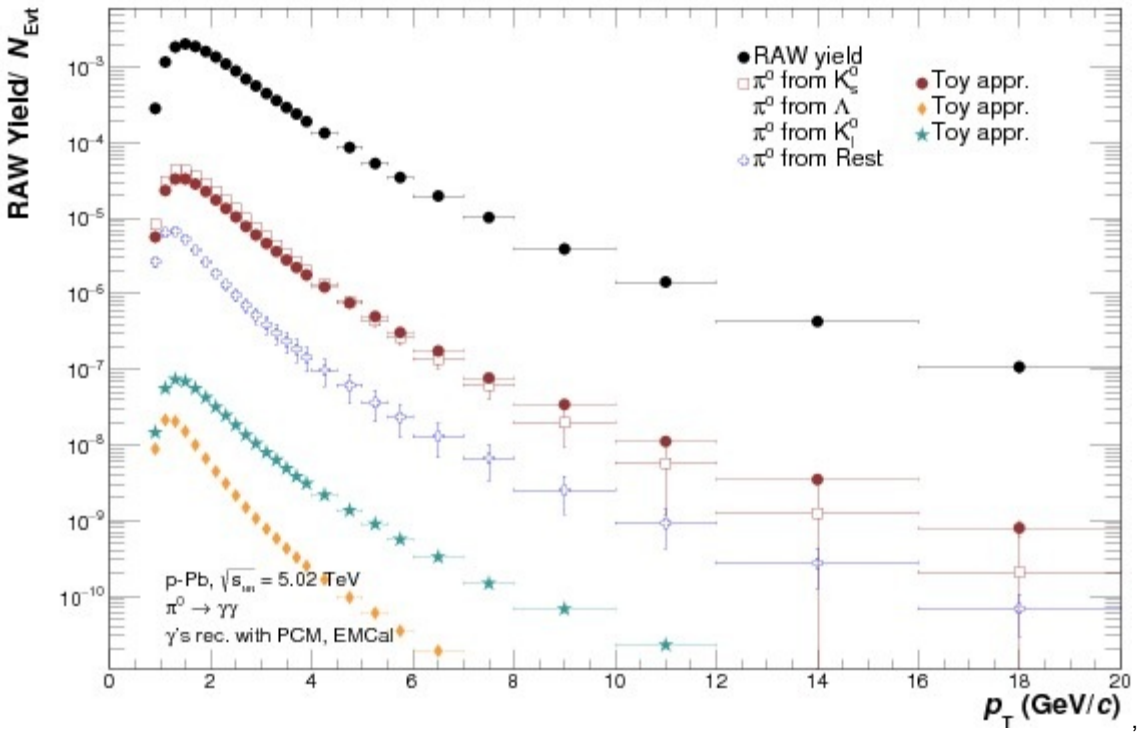
Pi0_TrueEffWithSecEffi_*



Pi0_data_EffectiveSecCorrPt_* & Pi0_data_RAWYieldSecPt_*



,



- Make sure the appropriate functions are used if you need to fix the secondary correction factors, these might depend on energy, collision system and reconstruction method, please try to adjust it such that you don't interfere with existing exceptions.
- Repeat the cocktail generation procedure, if the parametrizations are not good enough.

Preparing for the systematics running

CutStudiesOverview.C

- gives systematic uncertainty on cut variation with different sets of cuts.
- input: CutSelection.log + uncorected file, corrected file

Compilation of the final results for each collision system, energy and detector

ProduceFinalResultsV2.C

- gives corrected yield of each mesons with systematic uncertainties.

Merged Cluster Analysis

In addition to the already explained invariant mass based analysis methods using PCM and/or the calorimeters there is also the merged cluster analysis that is instead a particle identification analysis. It therefore relies heavily on the quality of the Monte Carlo simulation to obtain correct acceptance, efficiency and purity corrections.

The most important cut in the merged analysis is made on M02, the eigenvalue of the larger axis of an ellipse around the cluster. The cut on M02 allows the selection of unsymmetrical clusters with an elongation along one axis. Elongated clusters originate very likely from two particles that hit the calorimeter surface too close to each other and therefore the clusterizer incorporated the signal of both particles in one single cluster. Clusters in the EMCal start to merge above 10 GeV/c whereas in PHOS this happens much higher in pT due to the smaller cell size.

Due to the nature of this analysis, triggered data is of great importance to select mainly events that contain merged clusters. The EMCal triggers EMC7 and EGA are based on different energy thresholds and therefore enhance the yield above different transverse momenta. In the following example, pp 8 TeV data is used including its three triggers V0AND(MB), EMC7 and EGA.

The merged analysis is done with the following classes/macros in our repositories:

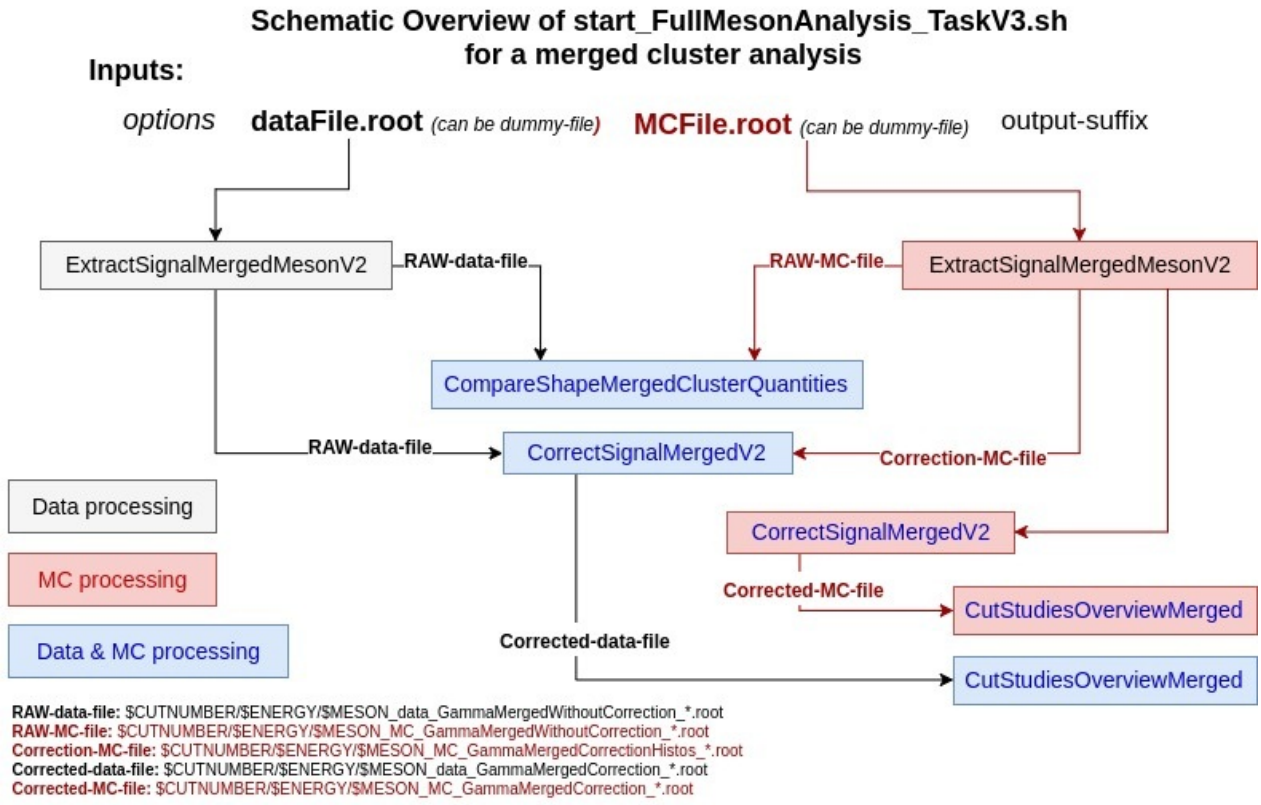
GRID: AddTask_GammaCaloMerged_pp/pPb, AliAnalysisTaskGammaCaloMerged

Afterburner: ExtractSignalMergedMeson, CompareShapeMergedClusterQuantities, CorrectSignalMerged

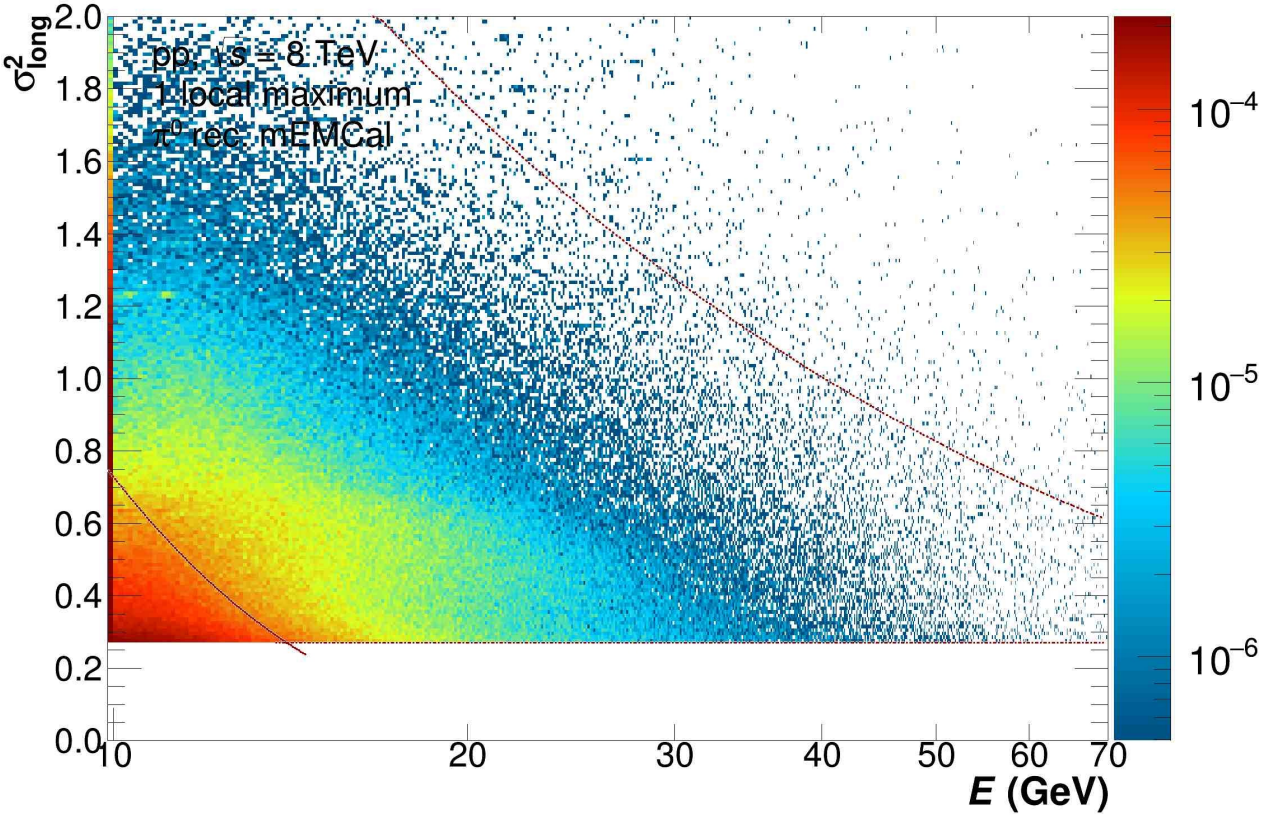
The task on the grid uses the same cut classes as the other calorimeter tasks, however, contrary to the standard calorimeter task two AliCaloPhotonCuts cutnumbers are defined. One cutnumber is used on all clusters and one to select the merged clusters. The merged analysis cutnumbers have the following format:

```
00010113_11111110670322000000_11111110670227000001_0163300000000000"
  event       cluster           merged cluster         meson
```

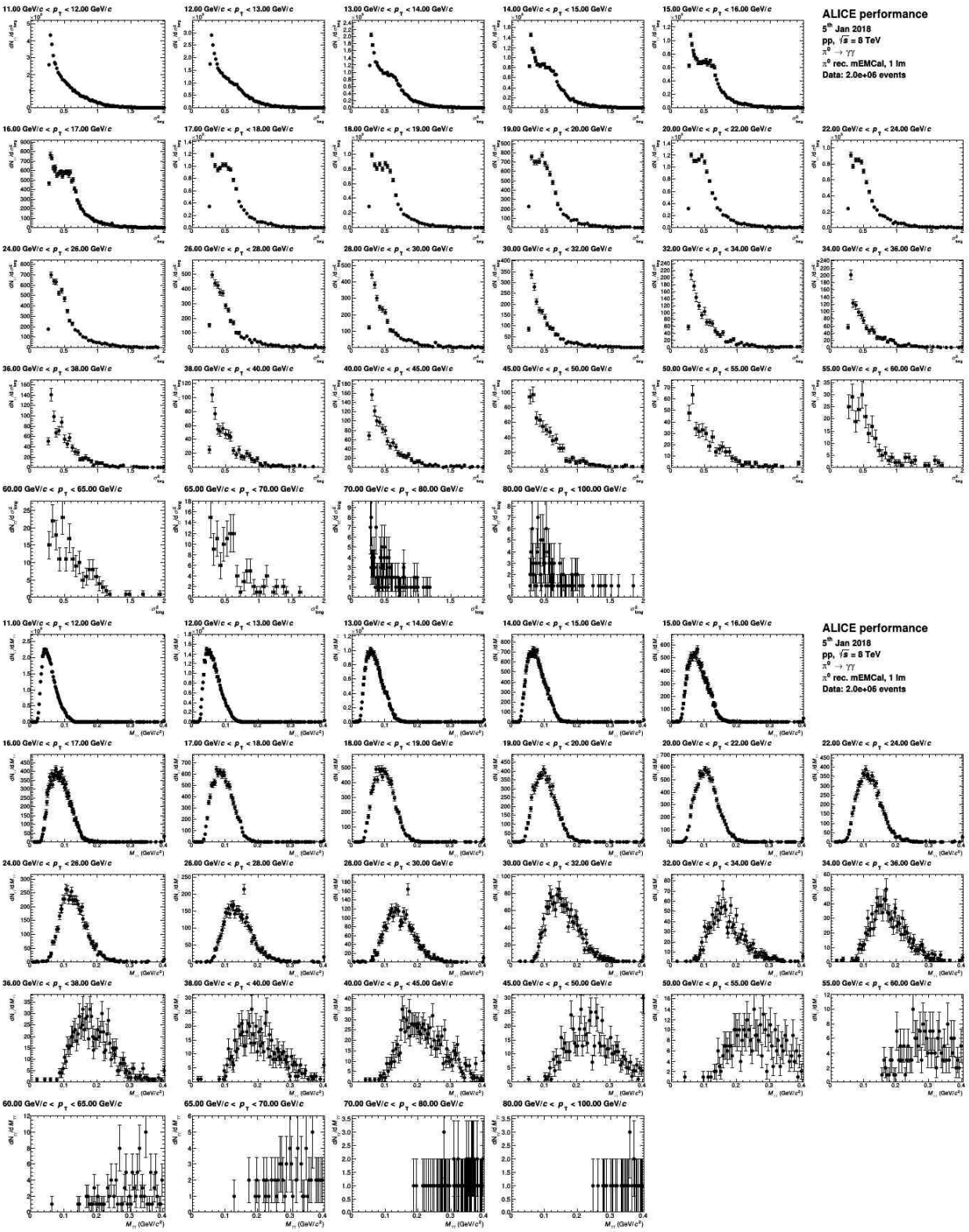
The task provides in the end an output file called GammaCaloMerged.root. This output is fed, as shown in the flow chart, into the AnalysisSoftware afterburners.



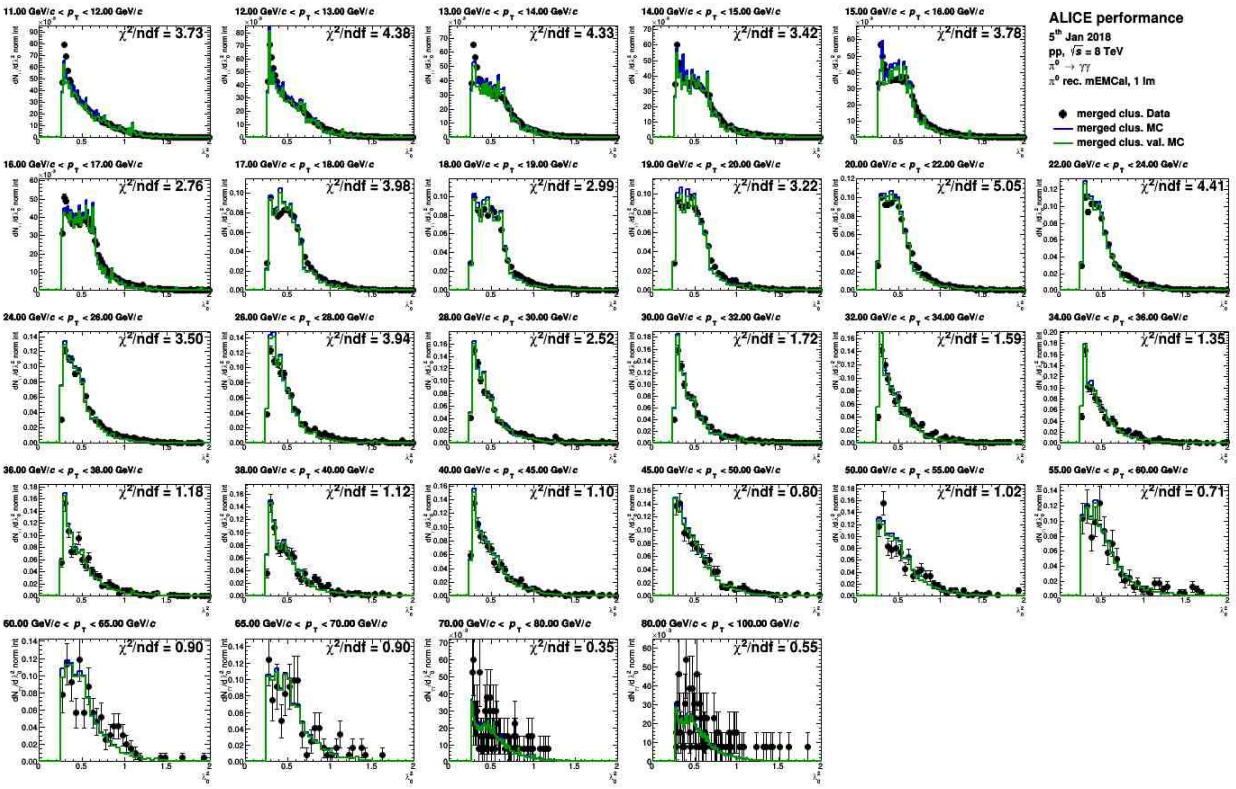
Running ExtractSignalMergedMeson produces several plots in the outputfolder `$CUTNUMBER/$ENERGY/ExtractSignalMergedMeson`. Control plots showing the energy dependend M02 cut in Pi0_data_EVsM02_AllAcceptedMesons are produced for data and MC, see:



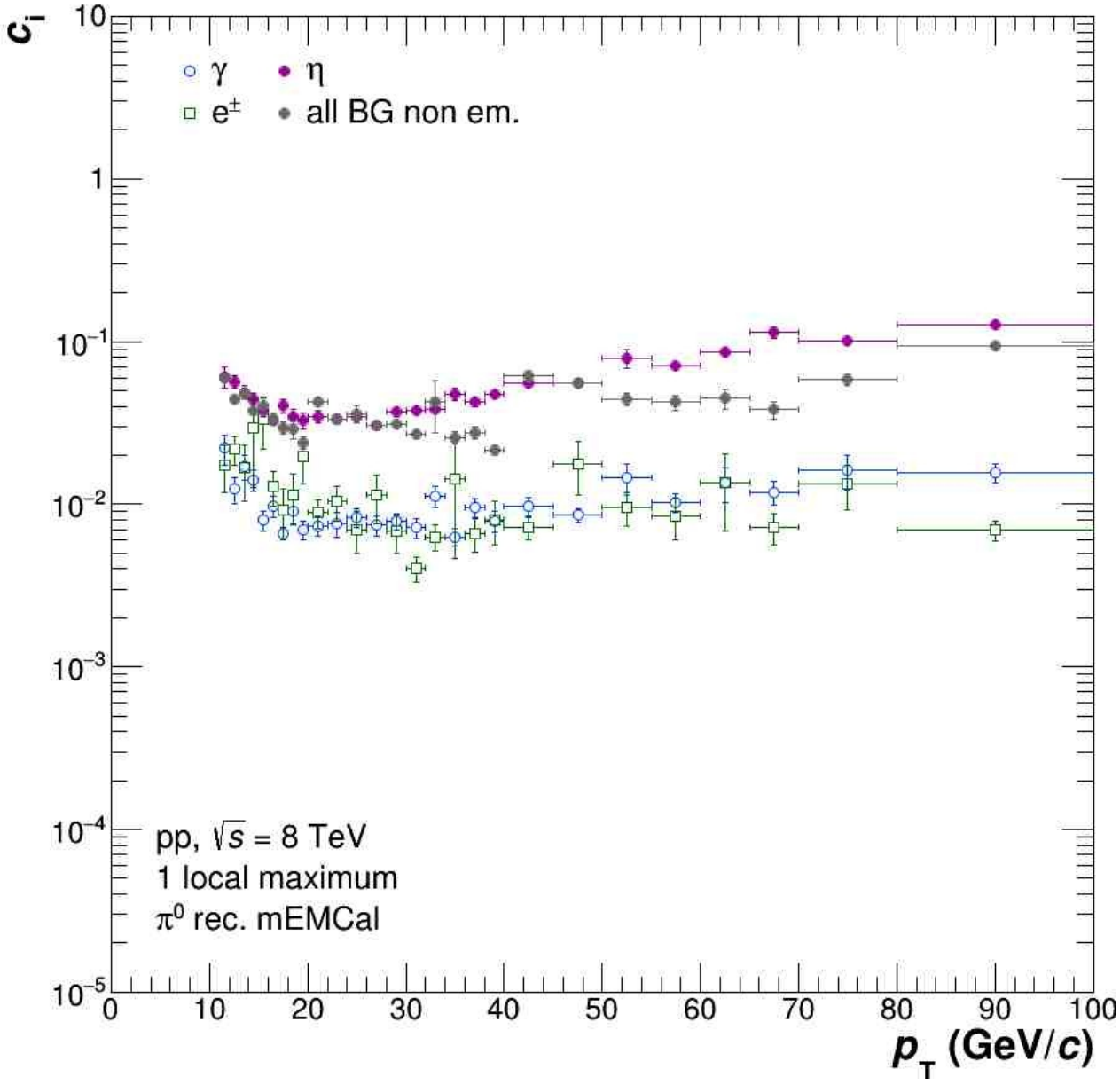
Furthermore, the M02 distributions for each pT bin are shown as well as the corresponding invariant mass distributions for the merged cluster pion candidates.

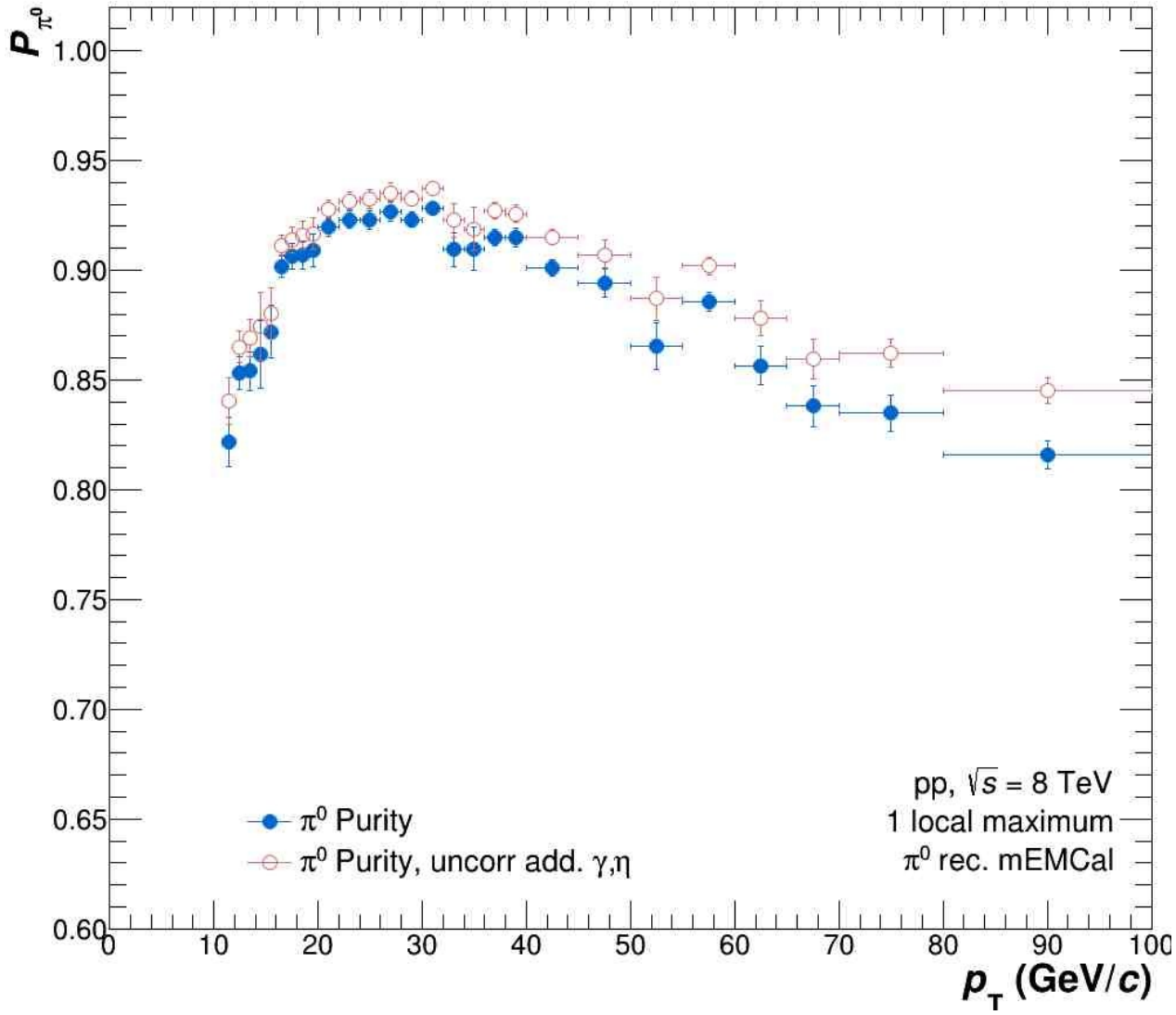


The second macro in the analysis chain (CompareShapeMergedClusterQuantities) makes necessary comparison plots between data and MC for different quantities (cluster energy, M02, number of cells per merged cluster and invariant mass). It can be seen that in Monte Carlo less cells per cluster will be present, however the invariant mass and M02 distributions agree quite acceptable, see:

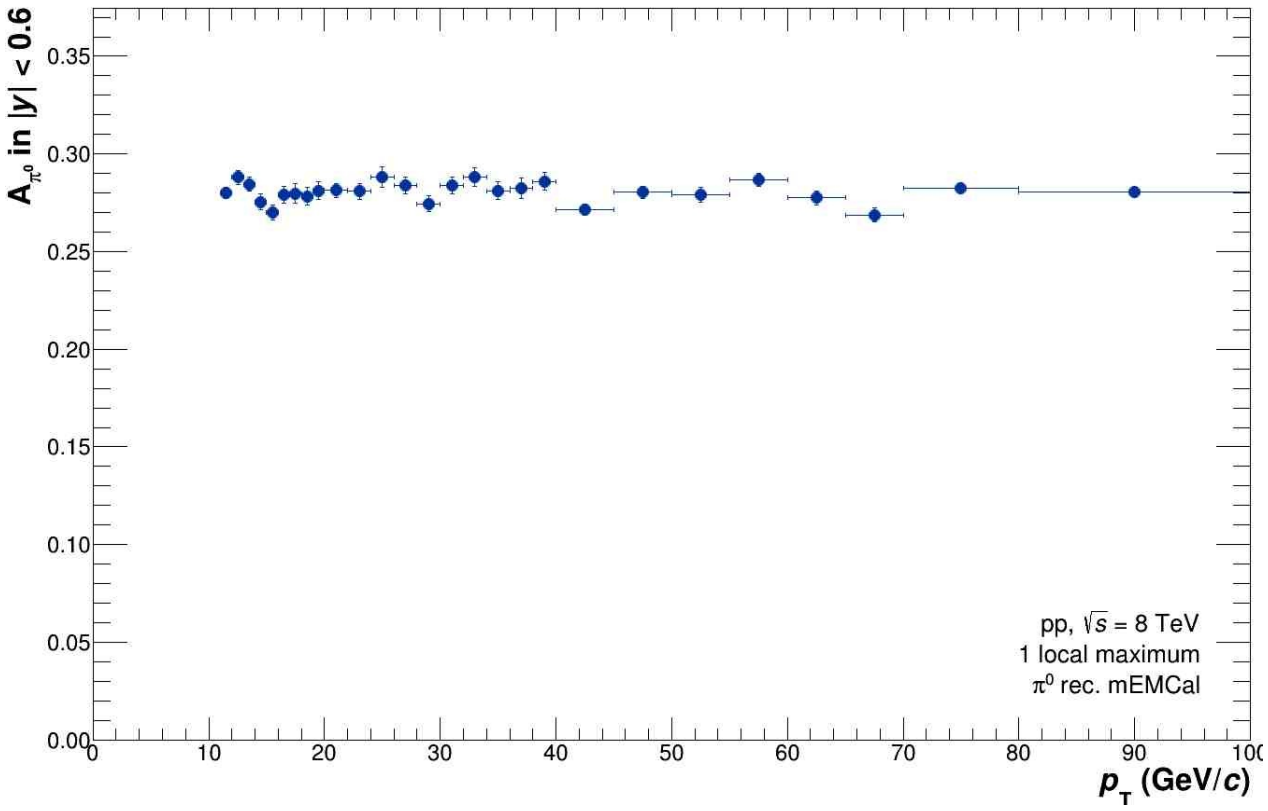


The correction of the raw yields from the merged clusters is done in CorrectSignalMerged. Here, acceptance, efficiency, secondary contamination and purity are determined from Monte Carlo and applied. In the following, the background contributions and corresponding purity correction are shown.

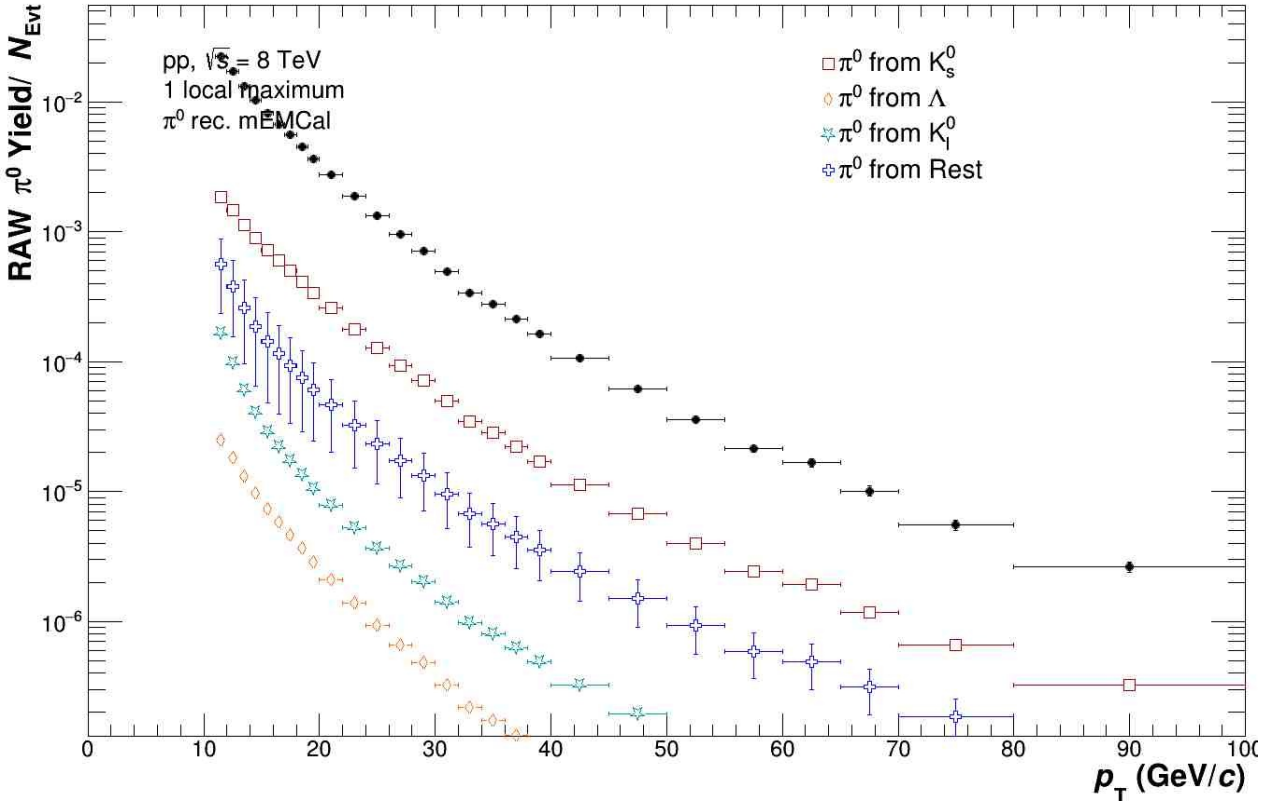

c_i
10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
γ
η
$e^{\pm}$
all BG non em.
pp, $\sqrt{s}$ = 8 TeV
1 local maximum
π^0 rec. mEMCal
0
10
20
30
40
50
60
70
80
90
100
p_T (GeV/c)



Furtheremore, the acceptance correction is given:



The secondary raw yields are shown below:



Systematic Uncertainties

The systematic uncertainty on the neutral meson measurements is evaluated, like in most analyses, by varying the selection cuts. These variations are chosen such that a reasonable deviation to the default selection criteria is tested.

For example we vary the rapidity selection of the reconstructed meson:

Default	Variation 1	Variation 2
-0.8 < y < 0.8	-0.6 < y < 0.6	-0.5 < y < 0.5

We then compare the corrected yield of each variation, as function of transverse momentum, to the corrected yield of the default selection cut. We always vary one cut at a time. In the case of our neutral mesons we can vary the cuts for photon and meson reconstruction. The full evaluation of the systematic uncertainty for a neutral meson analysis is handled in two parts, first by running all the different selection cuts on the grid, and then by the afterburner.

1. Setting up the cut configurations

The reconstruction of the meson depends on many cut selections and most of them are listed below.

Photon: PCM

- Material budget
- Min pt
- Chi2/ndf
- Psi pair
- TPC selection cuts
- Armenteros Podolanski variables (alpha and qt)
- ...

Photon: Calo

- Energy correction
- Material budget
- Cell timing
- Track Matching
- Min energy of cluster
- Min number of cells cluster
- Shower shape parameters
- ....

Meson: all

- Rapidity meson
- Opening angle photons
- energy asymmetry photons
- ...

We modify the AddTask accordingly

Default cut:

```
cuts.AddCut("80000113","1111141057032230000","01631031000000d0"); // 1 cell lead cell, 17mrad open
```

Variations 1:

```
cuts.AddCut("80000113","1111141057032230000","01631031000000a0"); // 1 cell lead cell, 0mrad open
```

Variations 2:

```
cuts.AddCut("80000113","1111141057032230000","01631031000000b0"); // 1 cell lead cell, 15mrad open
```

In the AddTask we would include these variations like this:

```
} else if (trainConfig == 21){ // default cutstring, 1cell distance lead cell
    cuts.AddCut("80000113","1111141057032230000","01631031000000a0"); // 1 cell lead cell
    cuts.AddCut("80000113","1111141057032230000","01631031000000d0"); // 1 cell lead cell, 17mrad open
    cuts.AddCut("80000113","1111141057032230000","01631031000000b0"); // 1 cell lead cell, 15mrad open
```

Remember to not put more than 4 or 5 cut variations in a single configuration, since that would consume too much memory on the grid.

2. Running all the cuts

After successfully running all the cut variations on the grid we need run the full meson analysis on all of them. Please check the following things:

- All the fits converge
- There is enough statistics available for all pt bins in both data and MC
- ...

3. Calculating the deviations to the default cut

After running the cuts we need to add the information to the CutStudies folder. For each set of variations we do it like(now an example for varying Alpha):

```
# # Alpha
echo -e "80000113_00200009327000008250400000_2444451041013200000_0163103100000010\n80000113_00200009327000008250400000_2
4444451041013200000_0163105100000010\n80000113_00200009327000008250400000_2444451041013200000_0163106100000010" > CutSele
ctionAlpha.log
    echo -e "Alpha\nLHC13bc\n3\nY\npPb5\nY\n/home/mike/2_pPb_EMC/0_analysis/170803_final_EMC/CocktailEMC_4Mio.root\n0.80
\nN\nY\nY" > answersAlpha.txt
    cp CutSelectionAlpha.log CutSelection.log
    bash start_FullMesonAnalysis_TaskV3.sh -dgammaOff bla eps < answersAlpha.txt
```

Note that this does not refit the mass peaks, it skips that part and runs the CutStudiesOverview on the variations and the default cut and stores it in the corresponding folder. This part is needed for the next macro that generates the systematic uncertainties.

After this procedure is done for all variations we use the following macro to calculate the systematic error:

```
FinaliseSystematicErrorsMETHOD_system.C
```

So for pPb it would be the following:

```
FinaliseSystematicErrorsConvCalo_pPb.C
```

In this macro we decide which of the cut variations to smooth and which contribute to the total systematic uncertainty.

4. Smoothing the deviations

Most of the deviations we observe should have a certain trend as function of transverse momentum and should not fluctuate too much bin by bin. For example think of the opening angle cut between the two photons, for any reconstruction method. In this case it would make no sense if you see the following trend:

bin 4	**bin 5**	**bin 6**	**bin 7**	**bin 8**
5% difference	7% difference	1% difference	11% difference	13% difference

We see that the deviation to the default cut increases rather linearly, but for bin 6 there is almost no difference observed. This is then most likely due to a statistical fluctuation. By interpolation one could put bin 6 at 9% systematic error. This is the concept of smoothing. Please be aware that this needs to be done with the physics in mind.

There is one contribution to the systematic uncertainty that is very rarely smoothed; the yield extraction. This quantity does and should fluctuate bin by bin, so it should not be smoothed.

5. Generating the final result

For generating the final result we use the following macro:

```
TaskV1/ProduceFinalResultsPatchedTriggers.C
```

Combination of Measurements

The combination of measurements is described in its latest version in Combination Notes pp 8 TeV, pp 2.76 TeV, from which all details about the procedures itself can be extracted. Further details about the combination of triggers within one method can be obtained from the respective analysis notes of the measurements, linked in Analysis Notes and Papers.

!!! Keep in mind that any combination of measurements is **HIGHLY** non-trivial and needs special care in all of its steps (uncertainties, correlations, bin-shifting, fitting, weights for combination, comparison with other measurements,...)! **!!!**

Correlation Coefficients of Statistical/Systematic Uncertainties

For the BLUE method, the statistical as well as systematic uncertainties of the measurements are needed as input for the combination (see Analysis Notes above).

In general, however, the statistical as well as systematic uncertainties may be correlated, for example combining PCM and PCM-EMCal but also for many other different cases (carefully think about any possibility how your measurements could be correlated in statistics+systematics -> general event cuts like SPD background rejection are the same for all measurements, although the correlation of systematics will be rather small, but still it exists).

For the case of different triggers being available for a reconstruction method, for example minimum bias and calorimeter triggers for EMCal/DCal/PHOS methods, the systematic uncertainties are found to be highly correlated in general - thus a proper consideration of correlation coefficients is already needed on the level of single methods/measurements.

An example configuration and how-to run the macro can be found here for the example of the neutral pion measurements for pp @ 8 TeV:

```
root -x -l -q -b ComputeCorrelationFactors.C\+\(\"input.log\"\,\"systems\"\,\"Pi0\"\,\"8TeV\"\,2\,\"eps\"\)
```

The macro needs an input log-file, for which you find an example configuration below.

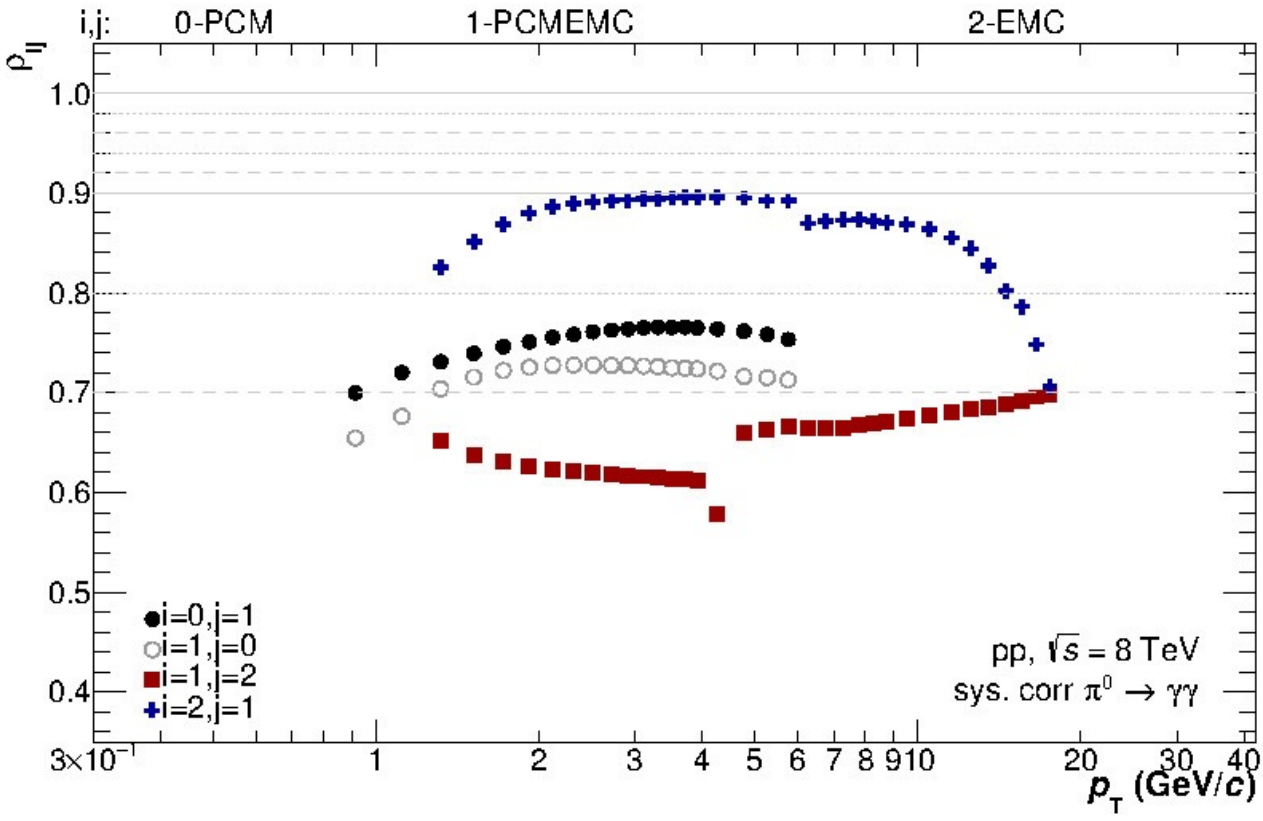
It takes the number of measurements and then the links to the systematic files, ***SystematicErrorAveragedSingle****, together with the method name, the number of bins and pT ranges.

Then, the single uncertainty sources are listed from the first method with respect to the second method, in how much % of the uncertainty is uncorrelated with respect to the latter method.

```
3
PCM 29 0.3 12.0 SystematicErrorAveragedSinglePCM_Pi0_8TeV_2017_11_03.dat
PCMEMC 40 0.8 35.0 FinalResultsTriggersPatched_PCMEMCAL/SystematicErrorAveragedSinglePCMEMC_Pi0_8TeV.dat
EMC 35 1.2 18.0 FinalResultsTriggersPatched_EMCAL/SystematicErrorAveragedSingleEMCEMC_Pi0_8TeV.dat
PCM PCMEMC YieldExtraction 95%
PCM PCMEMC PileupDCA 100%
PCM PCMEMC Material 50%
PCM PCMEMC dEdxE 75%
PCM PCMEMC dEdxPi 50%
PCM PCMEMC TPCCluster 25%
PCM PCMEMC SinglePt 50%
PCM PCMEMC Chi2 50%
PCM PCMEMC Qt 50%
PCM PCMEMC Alpha 25%
PCM PCMEMC BG 100%
PCM PCMEMC BGEstimate 100%
PCMEMC PCM YieldExtraction 95%
PCMEMC PCM Alpha 25%
PCMEMC PCM ConvPhi 100%
PCMEMC PCM ClusterMinEnergy 100%
PCMEMC PCM ClusterNCells 100%
PCMEMC PCM ClusterNonLinearity 100%
PCMEMC PCM ClusterTrackMatching 100%
PCMEMC PCM ClusterM02 100%
PCMEMC PCM CellTiming 100%
PCMEMC PCM ClusterMaterialTRD 100%
PCMEMC PCM Trigger 100%
PCMEMC PCM Efficiency 100%
PCMEMC PCM ClusterEnergyScale 100%
PCMEMC PCM ClusterTime 100%
PCMEMC PCM ClusterizationEnergy 100%
EMC PCMEMC YieldExtraction 95%
EMC PCMEMC OpeningAngle 100%
EMC PCMEMC ClusterMinEnergy 50%
EMC PCMEMC ClusterNCells 50%
EMC PCMEMC ClusterNonLinearity 50%
EMC PCMEMC ClusterTrackMatchingCalo 75%
EMC PCMEMC ClusterM02 50%
EMC PCMEMC CellTiming 50%
EMC PCMEMC Efficiency 50%
EMC PCMEMC ClusterEnergyScale 50%
EMC PCMEMC ClusterTime 50%
EMC PCMEMC ClusterizationEnergy 50%
PCMEMC EMC YieldExtraction 95%
PCMEMC EMC Material 100%
PCMEMC EMC dEdxE 100%
PCMEMC EMC dEdxPi 100%
PCMEMC EMC TPCCluster 100%
PCMEMC EMC SinglePt 100%
PCMEMC EMC Chi2 100%
PCMEMC EMC Qt 100%
PCMEMC EMC Alpha 100%
PCMEMC EMC ConvPhi 100%
PCMEMC EMC ClusterMinEnergy 25%
PCMEMC EMC Efficiency 50%
```

The correlation macro automatically reads in all the files and calculates the correlation coefficients for all common bins of the different methods.

It outputs a **.root* file with all the correlation factors as well as figures, examples shown below for the neutral pion pp@8TeV:



Not only systematic correlations can be calculated, the statistical correlation factors can also be determined using the macro and adjusting the list of parameters accordingly.

IMPORTANT NOTE

To use the correlation factors in your combination, make sure to add the cases to ***CombinePtPointsSpectraFullCorrMat***, example:

```
cout << "\n*********************************************************************************************" << endl;

cout << "loading systematic correlations..." << endl;

cout << "*********************************************************************************************\n" << endl;

corrFracPCM_PCM_PCMEMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "PCM_PCM-PCMEMC");
corrFracPCMEMC_PCM_PCMEMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "PCMEMC_PCM-PCMEM
C");
corrFracPCMEMC_PCMEMC_EMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "PCMEMC_PCMEMC-EM
C");
corrFracEMC_PCMEMC_EMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "EMC_PCMEMC-EMC");
corrFracPCM_PCM_EMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "PCM_PCM-EMC");
corrFracEMC_PCM_EMC[0] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems", mesonType, "EMC_PCM-EMC");

cout << "\n*********************************************************************************************" << endl;
cout << "loading statistical correlations..." << endl;

cout << "*********************************************************************************************\n" << endl;

if (GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCM_PCM-PCMEMC") != -10)
  corrFracPCM_PCM_PCMEMC[1] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCM_PCM-PCME
MC");
if (GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCMEMC_PCM-PCMEMC") != -10)
  corrFracPCMEMC_PCM_PCMEMC[1] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCMEMC_PC
M-PCMEMC");
if (GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCMEMC_PCMEMC-EMC") != -10)
  corrFracPCMEMC_PCMEMC_EMC[1] = GetCorrFactorFromFile(fCorrFactors,xValue[ptBin], "Systems_Stat", mesonType, "PCMEMC_PC
MEMC-EMC");
```

Combination of different Measurements

The combination of different measurements is performed in the macros ***CombineMeson...*** and ***CombineGamma...***. In the following examples are shown (many more macros are available in the *AnalysisSoftware* repository):

> CombineMesonMeasurements8TeV.C

```
root -x -l -b -q CombineMesonMeasurements8TeV.C\+\(\"$PCMfile\"\,\"$PCMEMCfile"\,\"$EMCfile\"\,\"$mergedEMCfile\"\
,\"$PHOSfile\"\,\"eps\"\,\"\"\,\"\"\,\"XY\"\,\"ComputeCorrelationFactors_pp8TeV/pp8TeV.root\"\,kTRUE\,kFALSE\)
```

The input files from PCM, PCMEMC, EMC, mergedEMC and PHOS need to be fed in, generated by ***TaskV1/ProduceFinalResultsPatchedTriggers.C***. Furthermore, the output mode for the figures ("eps") is specified as well as in which direction bin-shifting should be applied for the combined spectra ("XY"). The file containing the correlation factors for *pp@8TeV* needs to be hand over and some more plotting options are set with last two bools.

The macro performs the combination of all different input measurements and plots the final results, stored as *.eps* files as well as in *.root* files, see Combination Note pp 8 TeV which contains lots of the output generated by the combination macro.

> CombineGammaResultsPP8TeV.C

```
root -x -l -q -b CombineGammaResultsPP8TeV.C\+\(\"$PCMfile\"\,kFALSE\,\"\"\,kTRUE\,\"$EMCfile\"\,kTRUE\,\"$PCMEMCf
ile\"\,\"$cocktail\"\,\"C\"\,\"$corrFile\"\,kFALSE\,1.28\,kTRUE\,\"FitsPaperPP8TeV_2017_11_16.root\"\)
```

The input files from PCM, PCMEMC, EMC need to be fed in, generated by ***TaskV1/CalculateGammaToPi0V4.C***. Furthermore, the cocktail file is specified as well as the fits of the neutral meson spectra. The file containing the correlation factors for *pp@8TeV* needs to be hand over and some more plotting options are set. The number "1.28" specifies the nsigma to calculate upper limits on the direct photon spectra.

Useful functions

In this chapter a brief overview of the most useful functions in the framework is presented. These functions are used very often by the convenors of this tutorial and are always recommended to be used.

Plotting

1. The **basic plotting style** can be set in your macro via the following two lines. They take care of canvas colors, font styles, etc... and should generally always be used when plotting is intended in the macro.

```
StyleSettingsThesis();
SetPlotStyle();
```

2. Basic string functions:

```
TString dateForOutput   = ReturnDateStringForOutput(); //date in format: YYYY_MM_DD
TString collisionSystem = ReturnFullCollisionsSystem(energy); //i.e. "pp, #sqrt{s} = 7 TeV"
```

3. Basic **marker, color and size** settings for **different reconstruction methods** with `nameDet` being a string containing either one of: `"PCM", "PHOS", "EMCal", "PCM-PHOS", "PCM-EMCal", "PCM-Dalitz", "PHOS-Dalitz", "EMCal-Dalitz", "EMCal high pT", "EMCal merged", "PCMOtherDataset"` . **These colors should ALWAYS be used if the focus of a plot is to show multiple reconstruction methods!** These styles have been used in all recent publications where our group was involved.

```
Color_t  colorDet = GetDefaultColorDiffDetectors(nameDet.Data(), kFALSE, kFALSE, kTRUE);
Style_t  markerStyleDet = GetDefaultMarkerStyleDiffDetectors(nameDet.Data(), kFALSE);
Size_t   markerSizeDet = GetDefaultMarkerSizeDiffDetectors(nameDet.Data(), kFALSE);
```

4. Basic **marker, color and size** settings for **different center of mass energies** with a string containing the energy setting, i.e. "900GeV" or "8TeV". **These colors should ALWAYS be used if the focus of a plot is to show multiple center of mass energy measurements of the same method together!** These styles have been used in all recent publications where our group was involved.

```
Color_t colorEnergy       = GetColorDefaultColor("8TeV", "", "");
Style_t markerStyleEnergy = GetDefaultMarkerStyle("8TeV", "", "");
Size_t markerSizeEnergy   = GetDefaultMarkerSize("8TeV", "", "");
```

5. Set plotting styles for graphs, histograms, TF1:

```
DrawGammaSetMarkerTGraphAsym(dummgyTGrAsym , 2, styleLineJETPHOX, colorJETPHOX, colorJETPHOX, widthLinesBoxes, kTRU
E, colorJETPHOX,kTRUE);
```

6. Canvas setup:

```
TCanvas* canvas = new TCanvas("canvas","",200,10,1350,900);  // gives the page size
DrawGammaCanvasSettings( canvas, 0.08, 0.01, 0.01, 0.09); // sets margins (left, right, top, bottom)
```

   usually one directly sets the textsize in pixels for other plotting functions by using the canvas vertical pixel count (here 900) with:

```
Int_t textSizeLabelsPixel = 900*0.04;
```

7. Make and format legends quickly:

```
 TLegend* legendExample   = GetAndSetLegend2(0.12, 0.14, 0.45, 0.14+(0.04*expectedLinesInLegend), textSizeLabelsPixe
l, 2, "", 43, 0);
 legendExample->AddEntry(dummgyTGrAsym,"legend text","p"); // last argument: p (point), l (line), f (fill), e (error
), combinations possible, ie "pf", "lep", ...
 legendExample->Draw();
```

Calculation

Useful code

1. Cutting TGraphs to a certain x-range with while loop:

```
// remove points below 1.5 GeV/c
while(exampleGraph->GetX()[0] < 1.5)
   exampleGraph->RemovePoint(0);

// remove points above 20 GeV/c
while(exampleGraph->GetX()[exampleGraph->GetN()-1] > 20)
   exampleGraph->RemovePoint(exampleGraph->GetN()-1);
```