Tutorial session : running EPOS and RIVET

Johannès JAHAN (Ph.D. student) - Subatech / CNRS / Nantes Université

Quarkonia As Tools 2022 (9-15 January 2022)

With the (strong) support of :

Damien VINTACHE - Subatech / CNRS

Table of Contents

Program of this tutorial

In this tutorial, we will work with **EPOS 3.259** *(the last tuned version of EPOS 3*[1] *)* and **RIVET 3.1.5** *(last version of RIVET*[2] *released until now)*.

What you will do :

- write files to parametrise e^+e^- and pp/pA/AA simulations with EPOS
- run simulations with EPOS
- see the content of the main output formats from EPOS
- use a *(provided)* HepMC data file from EPOS to analyse it with RIVET
- produce plots with RIVET

What you won't do :

- produce HepMC data file with EPOS *(impossible with EPOS 3)*
- use EPOS 4 [3] *(not publicly available yet)*

All the instructions related to Docker will be written in grey, so that there's no confusion if you want to follow this tutorial with EPOS installed on you computer.

IMPORTANT : *all words in blue in these slides are hyperlinks to the corresponding webpages*

[1] K. Werner et al., *Phys. Rev. C* **89** (2014), 064903

[2] C. Bierlich et al., *SciPost Phys.* **8** (2020), 026

[3] K. Werner, et al., *Phys. Atom. Nucl.* **84** (2021), 1026–1029

Prerequisites

For simplicity, we will use **Docker** to run EPOS in a "container".
This way, you will benefit from a perfectly functional portable version, avoiding the necessity to install EPOS natively, with all its dependencies, on your computer.

Here are then the first necessary steps to follow, before to start this tutorial :

1. **Install Docker** *(for Linux, Windows or Mac)* by following instructions from :
   https://docs.docker.com/engine/install/

2. **Download** the **EPOS 3 image** and **EPOS_pp7T_hepmc.tgz** from :
   https://box.in2p3.fr/index.php/s/CgzXkqEDJPogcDc

You can then load this image, and check it worked (**epos3259** should appear), with :

```
> sudo docker load --input epos3259_latest.tar.gz
> sudo docker images
```

N.B. : normally, you would need to specify "`sudo`" before using any Docker command ; to get rid of this, do :

```
  > sudo groupadd docker
  > sudo usermod -aG docker $USER
  > newgrp docker
```
(for Linux ; otherwise disconnect+reconnect your session to apply changes)

(more information here : https://docs.docker.com/engine/install/linux-postinstall/)

Contents

What is EPOS ?

Event generators are programs made to compute models in order to simulate every step of a collision (e.g. EPOS, PYTHIA [4]...).

Advantages : - perfect detector, as final-state particles are all listed (no uncertainties)
- dynamical approach

*(indeed, there's always a **shadow in the picture** : one has to be careful on the applicability, and phenomenological approaches generally requires parametrisation)*

Energy conserving quantum mechanical approach, based on
Partons, parton ladders, strings,
Off-shell remnants, and
Saturation of parton ladders

Multi-purpose event generator based on parton-based Gribov-Regge Theory [5].
Can simulate with the same formalism any type of high-energy collision consistently :

$e^- + e^+$ $e^- + p$ $p + p$ $p + A$ $A + A$

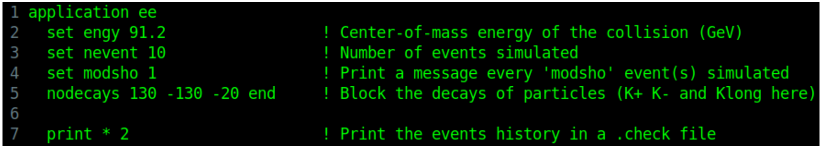
N.B. : *more details and references about EPOS in my talk on EPOS (Quarkonia As Tools 2022)*

[4] T. Sjöstrand et al., *Comput. Phys. Commun.* **191** (2015) 159-177
[5] K. Werner et al., *Phys. Rep.* **350** (2001) 93-289

Simulation of e^+e^- events with EPOS

To run simulations with EPOS, we need first to define a **.optns** file, containing all the information about the system.

First of all, create a dedicated directory in your folders (I will call it /**Tuto_EPOS**). Then, create a file "**ee91.optns**", and write in *(comments are indeed not mandatory)* :



```
application ee
  set engy 91.2                ! Center-of-mass energy of the collision (GeV)
  set nevent 10                ! Number of events simulated
  set modsho 1                 ! Print a message every 'modsho' event(s) simulated
  nodecays 130 -130 -20 end    ! Block the decays of particles (K+ K- and Klong here)

  print * 2                    ! Print the events history in a .check file
```

WARNING : never use tabs in **.optns** files !

To link your directory /**Tuto_EPOS** to the one in the Docker image :

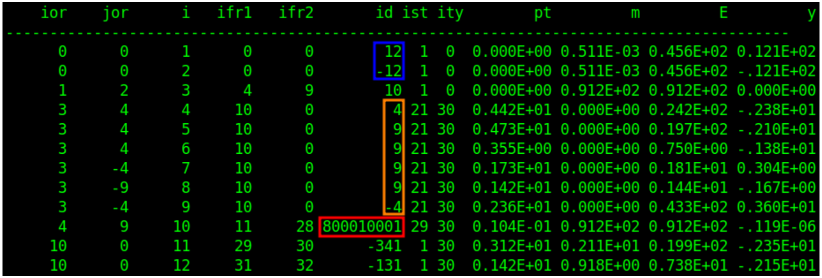
```
> docker run --rm -v /PATH/TO/Tuto_EPOS:/home/EPOS/optns:Z
                                   --entrypoint bash -it epos3259
```

It will open a new shell, in which you can now execute *(like if you had EPOS installed natively)* :

```
> ${EPO}epos ee91
```

.check output

You should now have in your repertory /**Tuto_EPOS** a new file called **z-ee91.check**.



```
   ior    jor      i  ifr1  ifr2       id ist ity       pt          m          E          y
------------------------------------------------------------------------------------------
     0      0      1     0     0       12   1   0  0.000E+00 0.511E-03 0.456E+02 0.121E+02
     0      0      2     0     0      -12   1   0  0.000E+00 0.511E-03 0.456E+02 -.121E+02
     1      2      3     4     9       10   1   0  0.000E+00 0.912E+02 0.912E+02 0.000E+00
     3      4      4    10     0        4  21  30  0.442E+01 0.000E+00 0.242E+02 -.238E+01
     3      4      5    10     0        9  21  30  0.473E+01 0.000E+00 0.197E+02 -.210E+01
     3      4      6    10     0        9  21  30  0.355E+00 0.000E+00 0.750E+00 -.138E+01
     3     -4      7    10     0        9  21  30  0.173E+01 0.000E+00 0.181E+01 0.304E+00
     3     -9      8    10     0        9  21  30  0.142E+01 0.000E+00 0.144E+01 -.167E+00
     3     -4      9    10     0       -4  21  30  0.236E+01 0.000E+00 0.433E+02 0.360E+01
     4      9     10    11    28 800010001  29  30  0.104E-01 0.912E+02 0.912E+02 -.119E-06
    10      0     11    29    30     -341   1  30  0.312E+01 0.211E+01 0.199E+02 -.235E+01
    10      0     12    31    32     -131   1  30  0.142E+01 0.918E+00 0.738E+01 -.215E+01
```

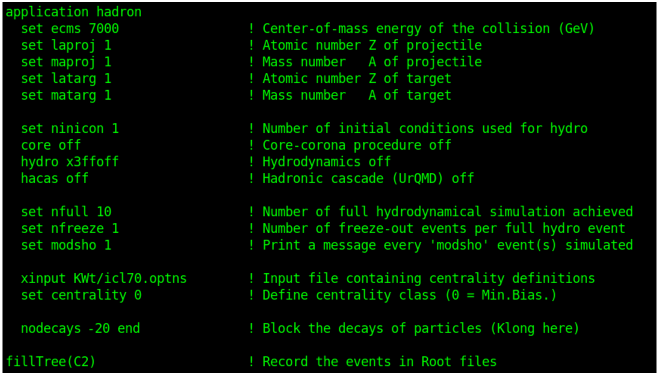
"**ior**"/"**jor**" : *parents of particle "**i**"*
"**ifr1**"/"**ifr2**" : *children of particle "**i**"*

You can clearly see the mechanism happening here :
the colliding e^- and e^+ annihilate into a photon, which is giving birth
to a chain of partons $c - g - ... - g - \overline{c}$ forming a string.

N.B. : *a lot of particle IDs in EPOS differ from PDG ; look at* ***idt.dt***

Simulation of pp/pA/AA events with EPOS

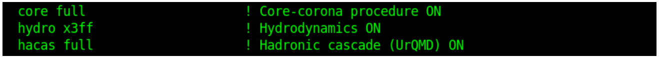
To run simulations for any hadronic system, you need a new file *(called **pp7T.optns** here)* :



```
application hadron
  set ecms 7000                ! Center-of-mass energy of the collision (GeV)
  set laproj 1                 ! Atomic number Z of projectile
  set maproj 1                 ! Mass number   A of projectile
  set latarg 1                 ! Atomic number Z of target
  set matarg 1                 ! Mass number   A of target

  set ninicon 1                ! Number of initial conditions used for hydro
  core off                     ! Core-corona procedure off
  hydro x3ffoff                ! Hydrodynamics off
  hacas off                    ! Hadronic cascade (UrQMD) off

  set nfull 10                 ! Number of full hydrodynamical simulation achieved
  set nfreeze 1                ! Number of freeze-out events per full hydro event
  set modsho 1                 ! Print a message every 'modsho' event(s) simulated

  xinput KWt/icl70.optns       ! Input file containing centrality definitions
  set centrality 0             ! Define centrality class (0 = Min.Bias.)

  nodecays -20 end             ! Block the decays of particles (Klong here)

fillTree(C2)                   ! Record the events in Root files
```

WARNING : again, no tabs in **.optns** files !

If you want to activate hydro evolution of the core + final-state hadronic cascades :
(hydro not recommended now, it would take a LOT of time)



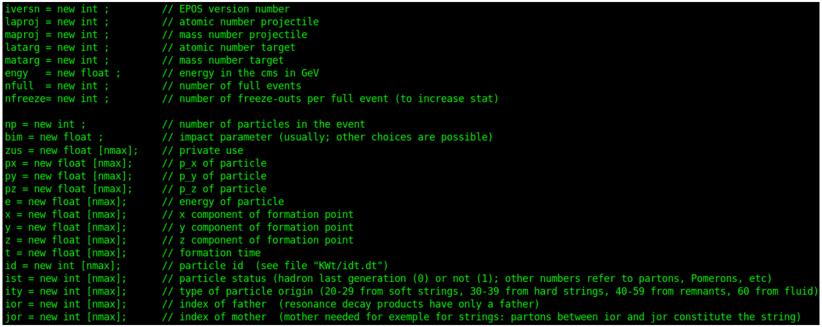
```
 core full                    ! Core-corona procedure ON
 hydro x3ff                   ! Hydrodynamics ON
 hacas full                   ! Hadronic cascade (UrQMD) ON
```

ROOT output

In order to produce a **ROOT file** containing the simulated events (based on ROOT 5.8, you have to add the flag "`-root`" before the name of the file to execute :

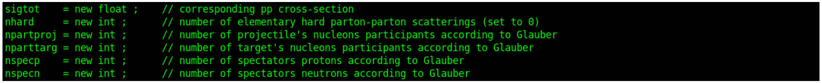
```
> ${EPO}epos -root pp7T
```

Here is the list of variables stored in the Trees of ROOT files produced with **EPOS 3** :



```
iversn = new int ;          // EPOS version number
laproj = new int ;          // atomic number projectile
maproj = new int ;          // mass number projectile
latarg = new int ;          // atomic number target
matarg = new int ;          // mass number target
engy   = new float ;        // energy in the cms in GeV
nfull  = new int ;          // number of full events
nfreeze= new int ;          // number of freeze-outs per full event (to increase stat)

np = new int ;              // number of particles in the event
bim = new float ;           // impact parameter (usually; other choices are possible)
zus = new float [nmax];     // private use
px = new float [nmax];      // p_x of particle
py = new float [nmax];      // p_y of particle
pz = new float [nmax];      // p_z of particle
e = new float [nmax];       // energy of particle
x = new float [nmax];       // x component of formation point
y = new float [nmax];       // y component of formation point
z = new float [nmax];       // z component of formation point
t = new float [nmax];       // formation time
id = new int [nmax];        // particle id  (see file "KWt/idt.dt")
ist = new int [nmax];       // particle status (hadron last generation (0) or not (1); other numbers refer to partons, Pomerons, etc)
ity = new int [nmax];       // type of particle origin (20-29 from soft strings, 30-39 from hard strings, 40-59 from remnants, 60 from fluid)
ior = new int [nmax];       // index of father  (resonance decay products have only a father)
jor = new int [nmax];       // index of mother  (mother needed for exemple for strings: partons between ior and jor constitute the string)
```

And the list of variables added in **EPOS 4** :

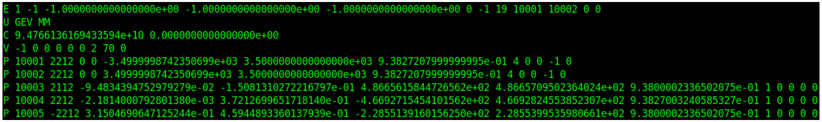


```
sigtot    = new float ;     // corresponding pp cross-section
nhard     = new int ;       // number of elementary hard parton-parton scatterings (set to 0)
npartproj = new int ;       // number of projectile's nucleons participants according to Glauber
nparttarg = new int ;       // number of target's nucleons participants according to Glauber
nspecp    = new int ;       // number of spectators protons according to Glauber
nspecn    = new int ;       // number of spectators neutrons according to Glauber
```

HepMC output

With **EPOS 4**, it is possible to produce **ASCII HepMC files** *(based on HepMC 2.06.09*[6] *)* which contain some part of the simulated events, making it compatible with **RIVET**.

To produce such files with **EPOS 4**, you shall add "`set ihepmc 1`" in the **.optns** file and add the flag "`-hepmc`" before the name of the file to execute.



```
E 1 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 0 -1 19 10001 10002 0 0
U GEV MM
C 9.4766136169433594e+10 0.0000000000000000e+00
V -1 0 0 0 0 0 2 70 0
P 10001 2212 0 0 -3.4999998742350699e+03 3.5000000000000000e+03 9.3827207999999995e-01 4 0 0 -1 0
P 10002 2212 0 0 3.4999998742350699e+03 3.5000000000000000e+03 9.3827207999999995e-01 4 0 0 -1 0
P 10003 2112 -9.4834394752979279e-02 -1.5081310272216797e-01 4.8665615844726562e+02 4.8665709502364024e+02 9.3800002336502075e-01 1 0 0 0 0
P 10004 2212 -2.1814000792801380e-03 3.7212699651718140e-01 -4.6692715454101562e+02 4.6692824553852307e+02 9.3827003240585327e-01 1 0 0 0 0
P 10005 -2212 3.1504690647125244e-01 4.5944893360137939e-01 -2.2855139160156250e+02 2.2855399535980661e+02 9.3800002336502075e-01 1 0 0 0 0
```

Detailed information about the structure of such files are given in HepMC 2 user manual (Sec.6).
*All the details about the data recorded in EPOS **.hepmc** files (especially for heavy-ion collisions) are given in **HepMC.txt** (might evolve with later developments of EPOS 4).*

In these files, we record the target + projectile, the final-state particles, as well as their parents (+grand-parents) when they decay with a lifetime $\tau > 10^{-20} s$
(+ some exceptions, i.e. J/Ψ, $\Psi(2S/3770)$ *and* $\Upsilon(1/2/3/4S)$*).*

$\Rightarrow$ enables to identify EM/weak feed-down contributions

N.B. : *feature NOT available here as we use **EPOS 3***

[6] https://hepmc.web.cern.ch/hepmc/

Contents

Prerequisites

For this tutorial, we will use the latest version released of **RIVET** [7], version 3.1.5.

To download a Docker image of RIVET version 3.1.5 (work the same for any version) :

```
> docker pull hepstore/rivet:3.1.5
```

Then, in order to link your /**Tuto_EPOS** directory to the one in the RIVET image (like we did for EPOS) :

```
> docker run --rm -v /PATH/TO/Tuto_EPOS:/work:Z
                        --entrypoint bash -it hepstore/rivet:3.1.5
```

Finally, you can test that RIVET works properly from the new shell opened with :

```
> rivet --version
```

N.B. : otherwise, you can define aliases in order to work from your directory, without opening a new shell, by following instructions from : https://gitlab.com/hepcedar/rivet/-/blob/release-3-1-x/doc/tutorials/docker.md

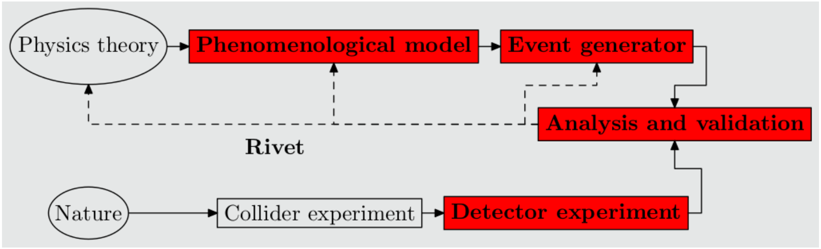
[7] https://gitlab.com/hepcedar/rivet

Robust Independant Validation of Experiment and Theory

RIVET is a ”software” based on C++ libraries, installed with different packages :

- YODA : Python libraries and classes used for analyses and histogramming
- HepMC : simulations recording and reading for analyses
- FastJet : recombination algorithms, mainly used for jet analyses

Purpose : offer a simple and standardised tool to automatise comparison between event generators simulations and experimental data



(C. Bierlich, 2019)

Robust Independant Validation of Experiment and Theory

RIVET is a ”software” based on C++ libraries, installed with different packages :

- YODA : Python libraries and classes used for analyses and histogramming
- HepMC : simulations recording and reading for analyses
- FastJet : recombination algorithms, mainly used for jet analyses

Purpose : offer a simple and standardised tool to automatise comparison between event generators simulations and experimental data

RIVET contains many analyses based on publications from many different experiments (data included), and develops thanks to contributions from the users community (from both experiments & theory).

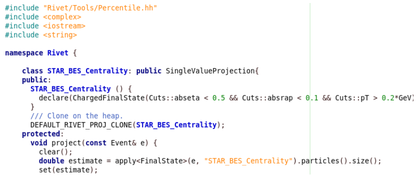
Advantages :

- provides huge and constantly growing library of data and analyses
- easy to handle (a lot of documentation + helpful reactive developers)
- don’t have to ”think about” the analysis details anymore

$\Rightarrow$ **RIVET is a very useful tool for us !**

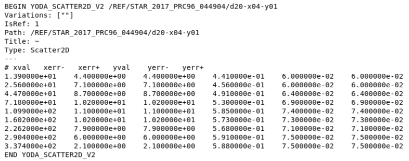
Composition of an analysis

Each analysis composed of 4 files :



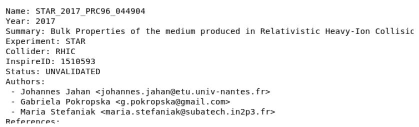
```
#include "Rivet/Tools/Percentile.hh"
#include <complex>
#include <iostream>
#include <string>

namespace Rivet {

    class STAR_BES_Centrality: public SingleValueProjection{
    public:
      STAR_BES_Centrality () {
        declare(ChargedFinalState(Cuts::abseta < 0.5 && Cuts::absrap < 0.1 && Cuts::pT > 0.2*GeV)
      }
      /// Clone on the heap.
      DEFAULT_RIVET_PROJ_CLONE(STAR_BES_Centrality);
    protected:
      void project(const Event& e) {
        clear();
        double estimate = apply<FinalState>(e, "STAR_BES_Centrality").particles().size();
        set(estimate);
```

ANALYSIS_NAME.cc
(analysis code)



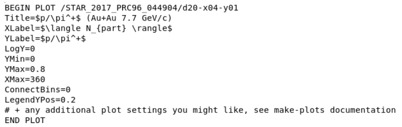
```
BEGIN YODA_SCATTER2D_V2 /REF/STAR_2017_PRC96_044904/d20-x04-y01
Variations: [""]
IsRef: 1
Path: /REF/STAR_2017_PRC96_044904/d20-x04-y01
Title: ~
Type: Scatter2D
---
# xval   xerr-   xerr+   yval    yerr-   yerr+
1.390000e+01    4.400000e+00    4.400000e+00    4.410000e-01    6.000000e-02    6.000000e-02
2.560000e+01    7.100000e+00    7.100000e+00    4.560000e-01    6.000000e-02    6.000000e-02
4.470000e+01    8.700000e+00    8.700000e+00    4.910000e-01    6.400000e-02    6.400000e-02
7.180000e+01    1.020000e+01    1.020000e+01    5.300000e-01    6.900000e-02    6.900000e-02
1.099000e+02    1.100000e+01    1.100000e+01    5.850000e-01    7.400000e-02    7.400000e-02
1.602000e+02    1.020000e+01    1.020000e+01    5.730000e-01    7.300000e-02    7.300000e-02
2.262000e+02    7.900000e+00    7.900000e+00    5.680000e-01    7.100000e-02    7.100000e-02
2.904000e+02    6.000000e+00    6.000000e+00    5.910000e-01    7.500000e-02    7.500000e-02
3.374000e+02    2.100000e+00    2.100000e+00    5.880000e-01    7.500000e-02    7.500000e-02
END YODA_SCATTER2D_V2
```

ANALYSIS_NAME.yoda
(experimental data)



```
Name: STAR_2017_PRC96_044904
Year: 2017
Summary: Bulk Properties of the medium produced in Relativistic Heavy-Ion Collisic
Experiment: STAR
Collider: RHIC
InspireID: 1510593
Status: UNVALIDATED
Authors:
 - Johannes Jahan <johannes.jahan@etu.univ-nantes.fr>
 - Gabriela Pokropska <g.pokropska@gmail.com>
 - Maria Stefaniak <maria.stefaniak@subatech.in2p3.fr>
References:
```

ANALYSIS_NAME.info
(information about paper, beam, author...)



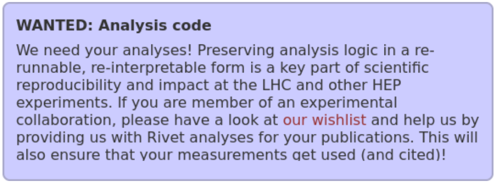
```
BEGIN PLOT /STAR_2017_PRC96_044904/d20-x04-y01
Title=$p/\pi^+$ (Au+Au 7.7 GeV/c)
XLabel=$\langle N_{part} \rangle$
YLabel=$p/\pi^+$
LogY=0
YMin=0
YMax=0.8
XMax=360
ConnectBins=0
LegendYPos=0.2
# + any additional plot settings you might like, see make-plots documentation
END PLOT
```

ANALYSIS_NAME.plot
(plotting options)

Analyses in RIVET

Catalogue of the analyses available in RIVET : https://rivet.hepforge.org/analyses.html

or by typing :
```
> rivet --list-analyses
```


WANTED: Analysis code

We need your analyses! Preserving analysis logic in a re-runnable, re-interpretable form is a key part of scientific reproducibility and impact at the LHC and other HEP experiments. If you are member of an experimental collaboration, please have a look at our wishlist and help us by providing us with Rivet analyses for your publications. This will also ensure that your measurements get used (and cited)!

Proposition of a list of relevant analyses to try during this tutorial :

- ALICE_2010_S8625980 (charged multiplicity)
- ALICE_2015_I1357424 (light hadrons)
- ALICE_2016_I1471838 (strange hadrons)
- ALICE_2012_I944757 (D mesons)
- ALICE_2017_I1645239 (Λ_c^+)

Running analyses with RIVET

We will run the analyses using a set of 50k events of pp collisions at $\sqrt{s} = 7$ TeV, obtained with **EPOS 4.328**, stored in the file **EPOS_pp7T_50k-evts.hepmc**.

IMPORTANT : preliminary version of EPOS 4, tuning not complete yet

In order to run any analysis of the catalogue with RIVET, you need to use (*you can "force" the analysis to run with an incompatible beam with* "`--ignore-beams`") :

```
> rivet DATA_FILE.hepmc -a ANALYSIS_NAME -o ANALYSIS_NAME.yoda
```

To get plots from the **.yoda** file obtained, you can type the following command :
*(it will create a folder called "**ANALYSIS_NAME**", containing all plots in **.pdf** and **.png**, with a generated URL where all the figures can be displayed.)*

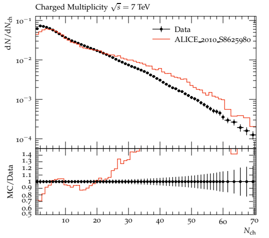
```
> rivet-mkhtml ANALYSIS_NAME.yoda -o ANALYSIS_NAME
```

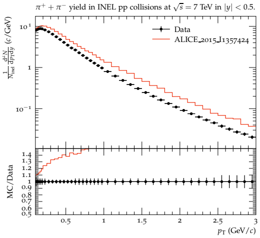
Finally, some analyses like **ALICE_2016_I1471838** require a centrality calibration. Hence, you need to run this calibration analysis (**ALICE_2015_PPCentrality** here) and call the obtained results after a "`-p`" flag when running your main analysis :

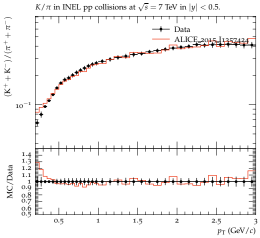
```
> rivet DATA_FILE.hepmc -a ANALYSIS_NAME:cent=GEN
                  -p CALIBRATION_NAME.yoda -o ANALYSIS_NAME.yoda
```

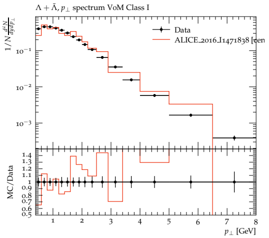
N.B. : *for more information on how to use RIVET, look at RIVET's ReadMe or C. Bierlich's "Hands-on session" (COST workshop 2019)*

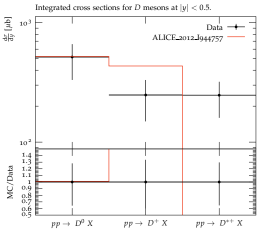
Some results you should obtain...

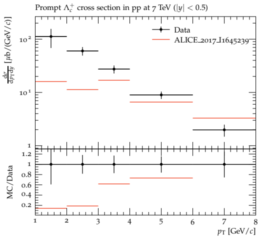












N.B. : *these data are only used for illustration ; they've been produced with a preliminary version of EPOS 4 (as already highlighted before), which explains why they don't fit well to the experimental results*

Thanks for your attention !

Hope you enjoyed it :)

All remarks, suggestions and further expectations will be welcome !

Contacts : johannes.jahan@subatech.in2p3.fr / klaus.werner@subatech.in2p3.fr