

ValidationsTools

Table des matières

General explanation :.....	3
KS curves.....	3
AE curves.....	4
0 – Launching the installation :.....	5
1 – Launching the ROOT files creation :.....	6
2 – Reducing size :.....	6
3 – Extracting values :.....	6
4 – Reference, comparison files :.....	6
5 – Checking ROOT files :.....	7
6 – Create some files and pictures :.....	8
7 – Comparing pValues for all added ROOT files :.....	9
8 – Generating AE pictures :.....	10
9 – Creating AE comparison pictures :.....	10
10 – Creating AE losses comparison pictures :.....	10
Annexe : ROOT scripts to modify.....	11
Annexe : paths definitions.....	12
Annexe : values for ROOT creation.....	12

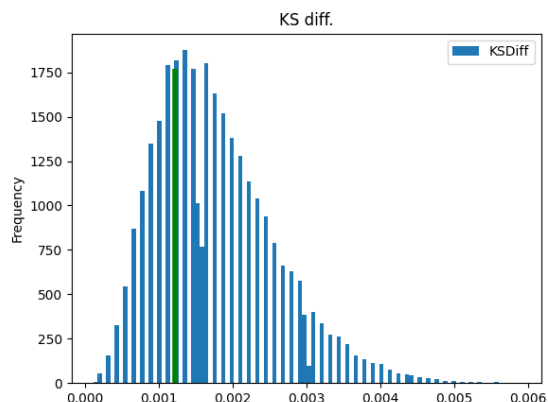
General explanation :

This repository get all the validations tools such as Kolmogorov-Smirnov (KS Tools) or AutoEncoders Tools (AE).

KS : talk about first dev (1 release vs 1 reference) to (1 reference vs lot of releases). pValues.

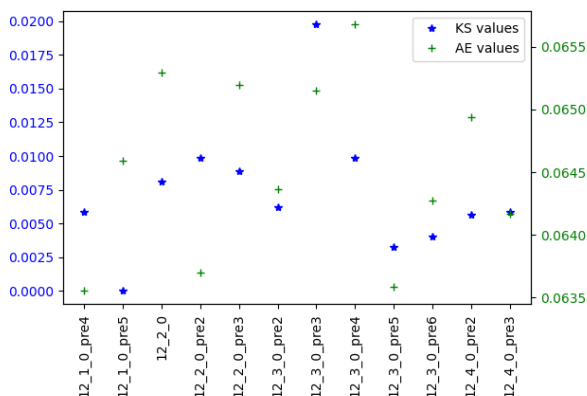
From the values of an histogram, we can construct an integrated cumulative curve. From this curve we can take the maximum of the difference between 2 consecutives values. Repeating this operation with a lot of ROOT files (200, 500 or 1000) we can construct a KS curve such here onto the right.

From this curve we can extract a pValue and try to obtain an idea of the validity of the histo and then the release.



AE : analyze on a reference and comparison with a lot of releases. The precedent explanation was made for one release vs 1 reference. Keeping the reference we can compare with a lot of releases (here about 12).

h_recEleNum : Comparison of KS vs AE values as function of releases.



From the ROOT files presented above we can train an autoencoder and try to predict the result of a given entry for one release. Doing this for a tenth of releases we can obtain a comparison, function of the releases, for the pValues or the differences at the end of the AE.

This is shown onto the left here, and extract an idea about the validity of one (or more) release(s).

One goal of those operations is from a reference release, to create a lot of ROOT files from this reference release. The set of the reference ROOT files are named as RRF or generated reference ROOT Files (GRRF).

Once you have created all those files (see 6), you will need to think about comparison.

In KS you have a reference release, and one or more release(s) to compare with. Into the first version of the KS tool, we have one « official » reference release (and a lot of created ROOT files from the same reference release). From those files we can compare a new release (in general called rel by comparison with the ref file) with the reference release and extract a lot of pictures for each histogram.

KS curves

First we generate an average curve bin to bin from the mean of all the histograms from RRF.

For each histogram we generate a graph with 3 curves representing the classical comparison of the histograms (ref vs rel) with the average curve added.

The new histo curve is in red, the reference one is in blue and the average curve in green.

We have 3 graphs more. The first one represents the cumulative curves of the 3 curves described above. A second graph representing the difference curve between the 2 (new/rel and reference/ref) cumulative curves. The last picture represents the difference curve between the new curve and the average curve.

With those 4 preceding pictures, we add 3 new pictures names as KS pictures and representing the Kolmogorov curves (see first picture above). For the first graph, we use all the references histograms and compare each accumulated curve with all the others.

For the second graph, we choose a random histogram as reference and compare its accumulated curve with each accumulated curve of all the others.

For the third graph, we took the histogram to be validated and compare its accumulated curve with each other accumulated curve of all the others histograms.

The last three curves are created with KS_Tools while the fourth others are created with a simple validation routine.

AE curves

For AE, we have a lot of difficulties. instead of KS, we have to search for a correct AE (number of layers, number of points for those layers, ...) and generate some pictures for a lot of releases (those placed into the DATA folder).

We can generate some comparisons with KS results (such as into the second picture above) as a function of a release, generate a prediction for each histo, and for each release. This can give some advice for the validity of the histogram and then for a release.

All thoses files are located into a folder named ValidationsTools which contains at least 5 folders :

- Doc : containing this documentation,
- ZEE_Flow : folder used for the creation of the ROOT files,
- KS_Tools : folder used for the KS pictures and files,
- AutoEncoder : containing the autoencoder tools and results,
- DATA : empty at the installation. Contain the « official » ROOT files, i.e. the releases ROOT files used for the comparison mentioned above.



later create a link to an explanation of the common files used for the ROOT files creation (CommonFiles/path...).

- CommonFiles : with the files which are used by all the tools, KS as AE.

Here is a summary of all the files/folders used for this tool.

CommonFiles : files used by all scripts

Doc : containing the documentation

default.py
pathsLLR.py
pathsLocal.py
pathsPBS.py
rootValues.py
sources.py
defaultStd.py
defaultDiff.py

DATA : empty at the installation. Contain the « official » ROOT files.

RESULTFOLDER : empty at the installation. Contain the «created» ROOT files (CRF).

ZEE Flow : folder used for the creation of the ROOT files

step1.py
step2.py
step3.py
step4.py
generateAE.sh
KScompare.sh
createFiles.sh
checkRootFiles.sh
createRootFiles.sh
reduceROOTFiles.sh
ReduceROOTSize.sh
extractValues.sh
reduceSize1File.py : reduce the size of the created ROOT files

KS Tools : folder used for the KS pictures and files

extractValues.py : create a file per histo with all values from the CRF
createFiles_v2.py : create a pValues file and a diffmax file for AE
extractNewFilesValues.py
KScompare.py : comparison between releases for KS
statpValues.py : pictures for pValues comparison between releases

AutoEncoder : containing the autoencoder tools and results

AEGeneration.py : generate some AE kernels, pictures and comparisons with KS results.

Check : folder used for the check of the ROOT files

checkRootFiles.py : create a check of the ROOT files (histos with pbms)
checkMapDiff.py : create maps for max ttl diff. Of the ROOT files

0 – Launching the installation :

- git clone <https://github.com/archiron/ValidationsTools> ValidationsTools
- cd ValidationsTools
- chmod 755 *.sh

then launch : **. installValidationsTools.sh newRelease**

This install the library (ChiLib), the release file env (cmsrel \$Release \$Release) and copy the ZEE_Flow/CCA or ZEE_Flow/LLR files (depending of the site you are working on – the CC or the

LLR site) onto the \$Release/src/Kolmogorov folder.

\$Release is the release you want working with. If you don't put any value this take a default release (CMSSW_12_1_0_pre5).

1 – Launching the ROOT files creation :

From the top of the Tools (i.e. ValidationsTools), then launch :

`. createROOTFiles.sh`

This script launch the creation of the ROOT files, using the step[1-4].py scripts.

You can use own but they have to be similar to the existing ones (see 11).

!! talk about the minimum size needed for nb_evts (~ 1000). 10 is too small and could be used only for development tests.

2 – Reducing size :

When ALL ROOT files are created, launch (always from the top folder) :

`. reduceROOTSize.sh`

this will reduce the size of the ROOT files (typically from 150/200 Mo to 1.5/2 Mo), keeping the name of the file. We only kept the used branches i.e. DQMData/Run 1/Info and DQMData/Run 1/EgammaV.

3 – Extracting values :

When all the ROOT files are created and reduced, we need to creates 1 file per histo with all the histo values for each ROOT file.

It is the first job of the extractValues[_init].sh scripts.

It can be launched with : `. extractValues_init.sh`

This create into the RESULTFOLDER (see 12) a lot of text files. All ROOT files are read, and then for each histo, the values of the histo curve is stored into an array, ROOT file after ROOT file.

Once we have all the ROOT files read, the array is stored into a text file dedicated to this histo.

The name is : RESULTFOLDER + 'histo_' + str(leaf) + '_' + '{:03d}'.format(nbFiles) + '.txt' where leaf is the name of the histo, nbFiles the number of the ROOT files (**WARNING** : it is **NOT** the same as NB_EVTS in rootValues.py – see 12).

Example :

histo_h_ele_zEff_200.txt
histo_h_ele_seedDphi2_200.txt

After the extraction of the values, if you want to work with AE, you will need to generate another file : branchesHistos_NewFiles.txt which contains the values of all histos for all the added ROOT files. The creation of this file is made with the same script as above which launch 2 python files : one for the extraction and another for this file creation.

4 – Reference, comparison files :

Talk about the files to add into the DATA folder. empty at the installation. Contain the « official » ROOT files, i.e. the releases ROOT files used for the comparison mentioned above.

The main goal of the KS tests is to validate a new release by comparing it to a reference. In order to simplify the process we first put those 2 files into the DATA folder. The reference file (named input_ref_file for the computation) is called with the sources.py file.

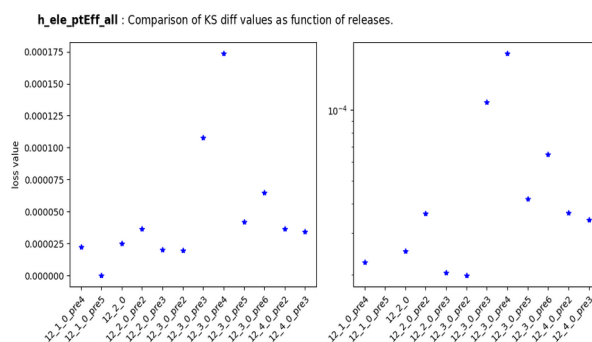
But, if we can make a comparison between a reference ROOT file and a new one, what's happen when we put a lot of files (say 10) of various releases. We always a reference but we can have a lot of new files and we can have an evolution of values with the releases.

Release
12_1_0_pre4
12_1_0_pre5
12_2_0
12_2_0_pre2
12_2_0_pre3
12_3_0_pre2
12_3_0_pre3
12_3_0_pre4
12_3_0_pre5
12_3_0_pre6
12_4_0_pre2
12_4_0_pre3

In the array on the left we have 12 releases. 12_1_0_pre5 is the reference so we can compare the 11 others with it.

An example is given with this picture where we have the maximum of the differences of the cumulative curve of a release with the reference release for the h_ele_ptEff_all histo.

For the reference release the difference is 0 and we can see some possible problems with 2 others releases (12_3_0_pre3 & 12_3_0_pre4).



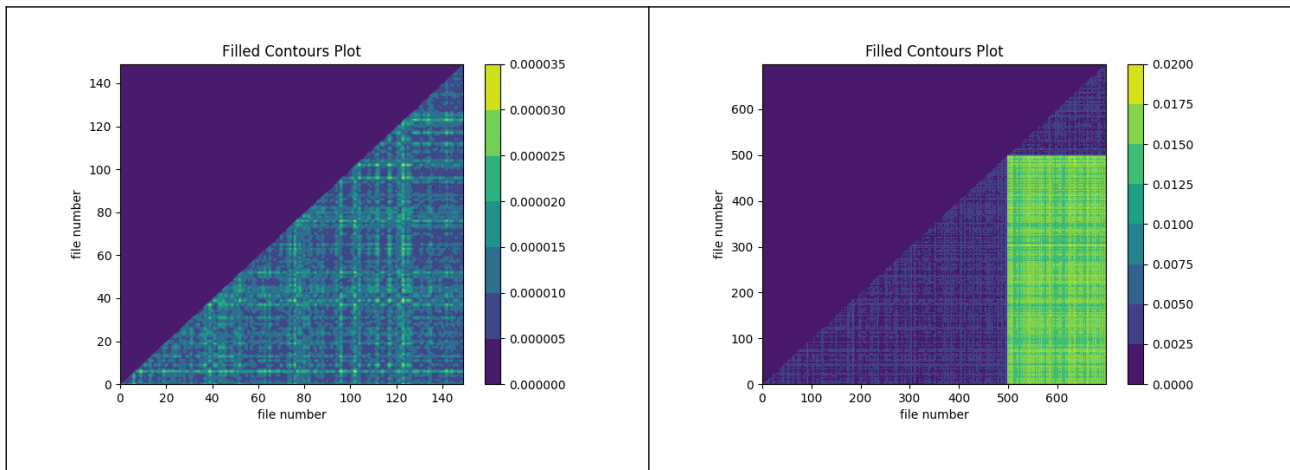
5 – Checking ROOT files :

When ALL ROOT files are reduced, launch (always from the top folder) :

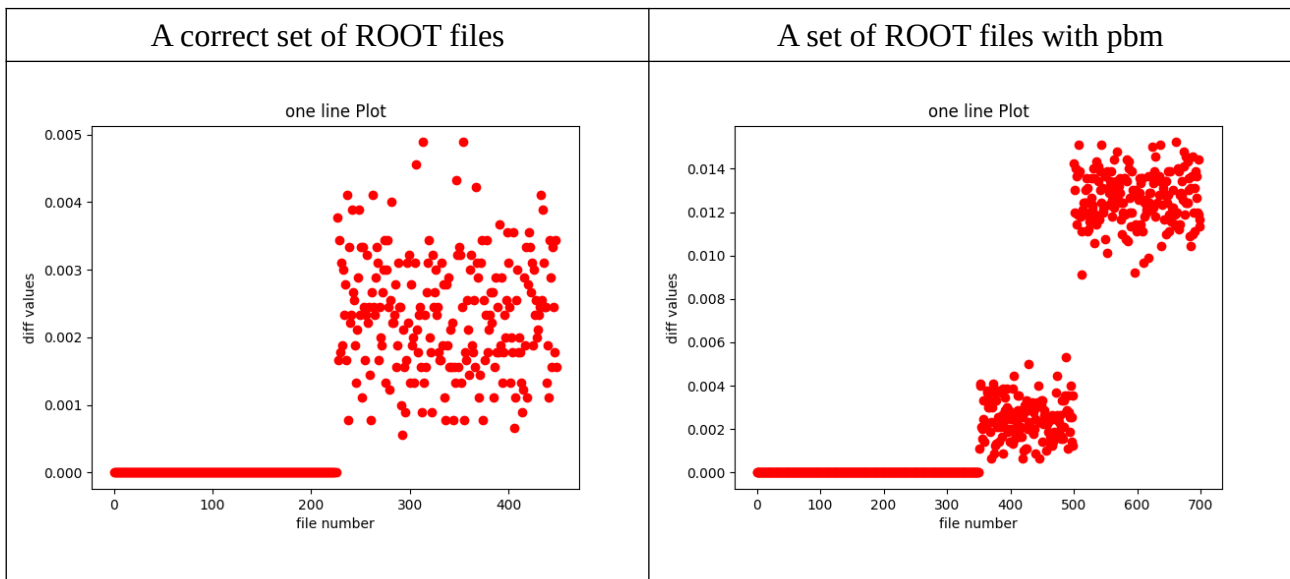
`. checkRootFiles_init.sh`

This will make a test of the ROOT files (those which are generated but also those which are added), and have an idea of some problem which can occur. The problems are generally a problem of a number of histograms.

A correct set of ROOT files	A set of ROOT files with pbm
-----------------------------	------------------------------



We can see on the maps above that on the left all the ROOT files are consistent (we plot for each histo the maximum difference between each ROOT file cumulative curve with each other curve) because there is no excess value. On the left there is a pbm with some histos as we can see with the colored map.



Here on the right we can see on a cut of the precedent right map that there is a clear step into the values while on the left there is no step.

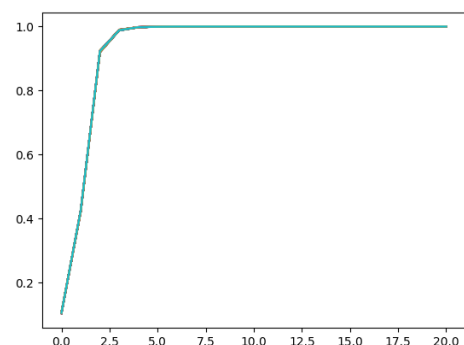
There is also some more tips displayed into a log file.

6 – Create some files and pictures :

When ALL ROOT files are created and values extracted, launch (always from the top folder) :

`. createFiles_init.sh`

Cumulative : for each histogram we extract from each ROOT file the values (see 6) and then create accumulated curves when it is possible. Those curves à normalized to the range [0, 1] and we display all the curves; it's a sort of new test because in case of problem, we will see some differences onto the curves.



KS_ttlDiff : For the first graph (KS_ttlDiff_1), we use all the references histograms and compare each accumulated curve with all the others and took the maximum of the difference.

For the second graph (KS_ttlDiff_2), we choose a random histogram as reference and compare its accumulated curve with each accumulated curve of all the others.

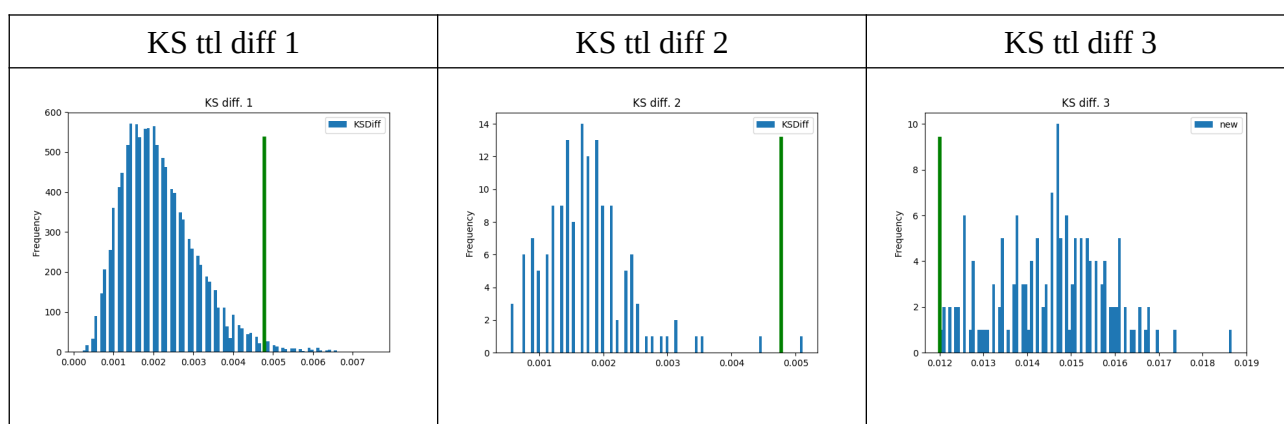
For the third graph (KS_ttlDiff_3), we took the histogram to be validated and compare its accumulated curve with each other accumulated curve of all the others histograms.

For each comparison, we take the maximum value of the difference.

For the 3 graphs, we took the max difference written above in the central box, and display it with the following color code :

if the value is included into the x-axis limits, the max. difference corresponding to the p-Value is displayed as a **green** line,

and if the value is lesser or greater than the x-axis limits, then the corresponding line is in **red**.



There is also the creation of numerous files with pValues for different release/histogram in order to use it later with the Comparison nor the AE generation.

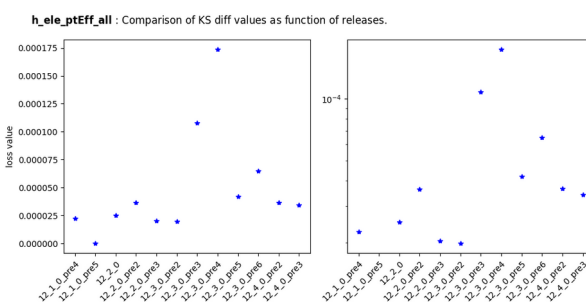
7 – Comparing pValues for all added ROOT files :

When ALL ROOT files are created and values extracted, launch (always from the top folder) :

`.KScompare_init.sh`

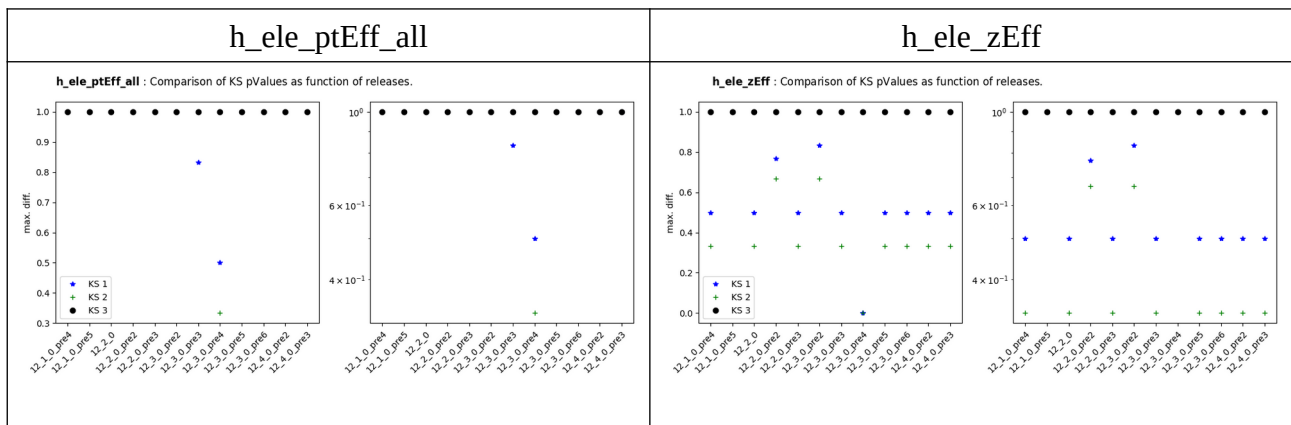
comparison_KS_values : As we made numerous files into the preceding step, we can compare each accumulated curve with the reference one (12_1_0_pre5 here). For all those curves, we extract the maximum of the difference as reported on the picture on the right (see the difference =0 when comparing the reference with itself).

We have 2 parts into the picture, the right one is the same as the left one but with a log vertical axis.

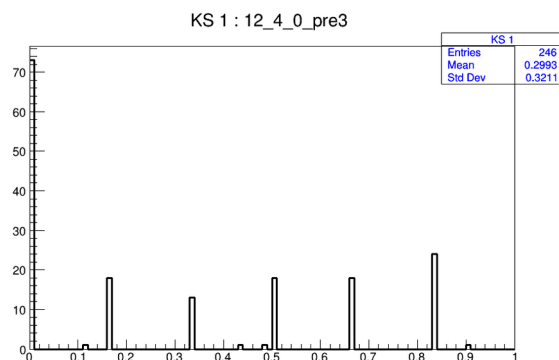


comparison_pValues : In a same way we can extract the pValues from the pictures like KS_ttl_Diff_X ($1 \leq X \leq 3$) and compare each other with the others releases. Compared to the precedent case we have now 3 curves of the pValues vs releases corresponding to one of the three cases.

Two examples are displayed on the table below for 2 histograms :



Talk about stat pValues (asked by Florian) with : KS_1_, KS_2_, KS_3_ Florian



8 – Generating AE pictures :

We now are looking for AE. After a choice of the number of layers and number of points per layer, we launch :

`. generateAE_init.sh`

talk about comparison with pictures.

9 – Creating AE comparison pictures :

We now are looking for AE. After a choice of the number of layers and number of points per layer, we launch :

`. resumeAE_init.sh`

talk about comparison with pictures.

10 – Creating AE losses comparison pictures :

We now are looking for AE. After a choice of the number of layers and number of points per layer, we launch :

`. compAELosses_init.sh`

talk about comparison with pictures.

ANNEXE

In that follow, we have the following notation :

al numeric values (int/float) are in **blue**.

Annexe : ROOT scripts to modify

Here we are talking about the step[1-4].py scripts. You can download the needed scripts onto the dmytro site ¹ and use the reqmgr into the left bottom of the page ².

In all stepX.py files, we need to replace the 2 lines :

```
from Configuration.Eras.Era_Run3_cff import Run3

process = cms.Process('SIM',Run3)
```

with :

```
from Configuration.Eras.Era_Run3_cff import Run3

if len(sys.argv) > 1:
    print(sys.argv)
    print("step 1 - arg. 0 :", sys.argv[0])
    print("step 1 - arg. 1 :", sys.argv[1])
    print("step 1 - arg. 2 :", sys.argv[2])
    print("step 1 - arg. 3 :", sys.argv[3])
    print("step 1 - arg. 4 :", sys.argv[4])
    ind = int(sys.argv[2])
    max_number = int(sys.argv[4])
else:
    print("step 1 - rien")
    ind = 0
    path = ""
    max_number = 10 # number of events

max_skipped = ind * max_number

process = cms.Process('SIM',Run3)
```

And also replace :

```
fileName = cms.untracked.string('file:step1.root'),
```

with :

```
fileName = cms.untracked.string('file:step1_' + '%0004d'%max_number + '_' +
'%003d'%ind + '.root'),
```

-
- 1 Such as https://dmytro.web.cern.ch/dmytro/cmsprodmon/workflows.php?prep_id=CMSSW_11_0_0_pre13_UPSG_Std_2026D41noPU-1575298095-ZEE_14
 - 2 https://cmsweb.cern.ch/reqmgr2/fetch?rid=ssawant_RVCMSSW_11_0_0_pre13ZEE_14_2026D41noPU_191202_155322_5298

And for all file name such as step2.root or other.

In the last script (usually step4.py) which generate the DQM file, we have to insert 2 lines between the dqmsave_step line such as :

```
part3 = '/RECO_' + '%0004d'%max_number + '_' + '%003d'%ind
process.dqmSaver.workflow = '/Global/' + 'CMSSW_X_Y_Z' + part3
process.dqmsave_step = cms.Path(process.DQMSaver)
```

Annexe : paths definitions

All those path (except RESULTFOLDER) must have to be joined to the local path where ValidationTools is i.e. pwd() =/ValidationsTools. This is also named as **top folder**.

LOG_SOURCE="ZEE_Flow/CMSSW_12_1_0_pre5/src/Kolmogorov"

LOG_SOURCE : path used to create the ROOT files. Need to keep a cmsRun env.

LOG_OUTPUT="/sps/cms/chiron/TEMP/"

RESULTFOLDER="/sps/cms/chiron/Validations"

RESULTFOLDER : path where you want the created ROOT files are located. It also contain the text files for each histo.

LOG_KS_SOURCE="KS_Tools/"

LIB_SOURCE="ChiLib/"

COMMON_SOURCE="CommonFiles/"

DATA_SOURCE="DATA/"

CHECK_SOURCE="Check/"

LOG_AE_SOURCE="AE_Tools/"

Annexe : values for ROOT creation

For the creation of the ROOT files we need python files with physics and for those files we need some values.

In general we need a lot files and not only one. So, those file went from **Nbegin = 0** to **Nend=200** or **1000**.

For a lot of tests it can be nice to have smaller values such as :

Nbegin = 20

Nend = 23

Inside the python files we need to know about the number of events to be used. Most of the « official » runs use **NB_EVTS = 9000**. For some rapid tests we can use :

NB_EVTS = 10

but for tests with values expected, we need NB_EVTS = 1000.