Manual for the tool EfficiencyMap_Creator

Federico Ambrogi

Institut für Hochenergiephysik, Österreichische Akademie der Wissenschaften,
Nikolsdorfer Gasse 18, 1050 Wien, Austria

March 29, 2016

1 Quick Running Instructions

Copy the folder anywhere in your local pc or heplx or wherever you want (if you are allowed to install MG5 and MA5). Please note that right now all the intermediate output are saved so it will require few hundred MB of disk space (or, erase some of the SLHA files from the slha input folder).
The script will not create folder outside its main directory, so you can easily delete the output.

Note: MA5 requires pyroot to be installed.

1.1 Instructions for a quick run

Open **EM_Ingredients.py** and check the following parameters:

Output_Dir	= ' your choice '	name of Output Folder
SLHA_InputDir	= Home_Dir + '/Input/slha_folder_small'	SLHA files folder
Installation_Switch	= 'ON'	will install MG5 and MA5
'NEVENTS'	: '500'	N Events at parton level

All the other parameters are irrelevant and will effect only the content of the samples, not the production chain.
Now, call:

```
python EM_Baking.py
```

It will ask for your user name; it should work out of the box.

I advice giving a first quick run to have an output in front of you, to easily understand what you read through the more detailed section.

2 Detailed Description of the Code

After unzipping the folder, you will find a few scripts and two sub-directories. The first is called `Programs_Archives` and contains the tarballs of MG5 and MA5 plus the cards needed for their correct installation. The second is called `Input` and contains two template cards for MG5 and Pythia.

The main code is called `EM_Baking.py`.It contains the calls to all the other auxiliary scripts that contain the utilities needed (for example the scripts that installs the softwares if needed, the parser of the EM results etc. , that will be described later):

- `EM_Ingredients`
- `Baker_Assistant`
- `Cards_Producer`
- `Software_Installator`
- `MA5_Output_parser`

Essentially the main code `EM_Baking.py` loops over all parameters chosen by the user and produces efficiency maps in a SModelS friendly format.

The folder `Input` contains the two MG5 and Pythia template cards that will be replaced by the values of the parameters chosen by the User (see `Em_Ingredients.py` description) , the `mg5_configuration.txt` that simply forbids the automatic opening of the web browser of the User after the production of LHE file (this card is automatically copied inside the MG5 User's directory), and two sub-folders called `MG5_Process_Cards` and `slha_folder_small`.The first contains the cards used by MG5 to build the generation processes (ex. T2tt + 1 ISR), the latter contains the SLHA files that the user wants to produce.

2.1 Em_Ingredients.py

All the input parameters are defined inside `EM_Ingredients.py`:the variables that should be defined by the user (i.e. that depend on the particular choices of the EM the user wants to produce) are stored inside this module and imported in the various sub-codes . The User does not have to modify anything else than this input card.

- `Output_Dir`: name of the output folder that will contain the file produced by the script.
- `SLHA_InputDir`: folder containing the SLHA file to be processed. The SLHA name have to be in the form `txName_M1_I1_D1.slha`or `txName_M1_D1.slha`,where `M1` stands for the mother mass, `D1` for daughter mass and `I1` for the intermediate particle mass; please go to the last section for comments. So for now, I assume this folder already exists and contain the SLHA I want to process. I think it is a nice idea to keep this folder somewhere

and erase the SLHA that have correctly been processed (i.e. the ones that have produced a correct EM output), so that User can keep track of the SLHA that she/he needs to rerun .

- `MA5_tar,MG5_tar:` Names MA5 and MG5 tarball included in this version of the script. This will be used in order to install the two softwares, if `Installation_Switch` is NOT set to 'OFF'. For any other value of `Installation_Switch,` the code will proceed to the installation of the two softwares.

  If the user wants to use his own installations he must set the paths `MG5_Dir`and `MA5_Dir`and the corresponding version of the codes. Installing the two codes takes around 5-10 minutes, so if one wants to save a bit of time, he can skip this step by using his own installations.

- `MA5_Analyses_List:`is the list of analyses we are interested in; in principle MA5 receives as input a card with only the analyses that we want to process rather than all the analyses implemented in the PAD (the database of all the MA5 implemented analyses), but the instruction I have in order to use this option do not work (I need to ask MA5 dvelopers). So right now, MA5 runs over all the analyses in the PAD but it is time consuming and not useful. After the complete MA5 process, I select only the ones that I choose here in this list. The name of the analyses should follow the MA5 convention; this will be explained in detail in when I will talk about the script `MA5_Output_parser`.

- `MG5_Pythia_Params` is a dictionary containing the core of all the physical parameters used for the sample production (by MG5 + Pythia). By default the MG5 process assumes the creation of one ISR jet.

  All the parameters chosen here will replace the default parameter inside the MG5 and Pythia template; so, apart from the processes list, the dictionary key corresponds to the string to be replaced inside the template card with the value of the key itself (i.e. the code looks for the string 'qcut' inside the pythia template card and will replace the string with the value `MG5_Pythia_Params['qcut']`).

2.2 Software_Installators.py

This utility simply installs MG5 and MA5 , in particular the call of the function `Install_MG5_MA5` proceeds to the installations and it return the path of the folder where the softwares have been installed or where the User has his own local installations (paths are needed by the main code).

Note for those familiar with MA5 :
if you install fastjet, the code will break for an error that does not make any sense to me. (it is a c++ complaint about a missing '}' or another MA5 issue not very clear). I asked Benjamin and he said that the installations of these tools IS NOT necessary anymore.

You can try and reproduce the error by uncommenting the corresponding line inside this script, and see how/why the code breaks.

2.3 Cards_Producer.py

This scripts simply produces the cards used as input by MG5 and MA5. In particular, MG5 needs three cards:

- the run_card, which is produced by reading the template and substituting the values listed in EM_Ingredients.py
- the param_card, which is actually the SLHA file used as input
- the pythia_card , same as the run_card

These three cards are produced by the function `MG5_Pythia_cards_producer;`the other two cards produced are called `MG5_commands_file.txt`and `MA5_commands_file.txt.`These are needed for the call of the two softwares, for example

```
./bin/ma5 -R MA5_commands_file.txt
```

.
In particular these two cards contain the list of commands that the user has to pass in the shell to execute the programme. The parameters that change in the two commands file are the process name in the case of MG5, and the HEP file as input in the case of MA5.
Please note that the commands needed to run these softwares change wrt the version of the softwares, so this commands might not work with a different version.

In principle MA5 needs also the recasting_card.dat as an input (the card selecting the analyses that the user wants to use), but this does not work at present, and will be implemented later as I have already mentioned before.

2.4 Baker_Assistant.py

This is the utility containing the most important function used by the main script. Each function is commented inside the code and should be easy to understand.

2.5 MA5_Output_parser.py

This script simply re-arrange the MA5 output in a SModelS friendly format. It produces as an output a txt file named as the ‘ MA5_EM_Official_region.dat ’ where Official_region is the name given by CMS or ATLAS.
One important thing I would like you to notice is the dictionary of Analyses and corresponding signal regions on top of the script (I will discuss it also in the last section).

2.6 EM_Baking.py

Also in this case the code is quite commented.

Recap of the steps:

1. input your user name (this step will be deleted)
2. check if the slha input directory exists; if not, the code breaks
3. determine the path of the MG5 and MA5 installations (If given by the user or if installed brand new)
4. start of the generation: loop over the MG5 processes
5. creation of the Output folder (will contain the EM and the partial output that after the test phase will be removed)
6. run MG5 and move the output to the correct user chosen output folder
7. run MA5 and move the output
8. Parsing of the MA5 analyses output and creation of the EM files
9. Saving the info about the production in the file 'EM_Baking_LogFile.txt'

3 To be Implemented

Here I describe briefly the things that I have already in mind to implement and to improve.

- **SLHA Creator** : the code right now works under the assumption that there exists a folder containing the SLHA one wants to process, and the SLHA are named under the convention *txName_Mother_Intermediate_Daughter.slha* or *txName_Mother_Daughter.slha*. One missing part is the creation of the SLHA itself ; since the SLHA files are small txt files and they do not occupy a lot of space, my suggestion is to keep them already in a tarball and just copy the ones that we need to the SLHA input folder when creating the map.

  Once the complete set of SLHA is done, it is then only a matter of selecting the grid that we need for the specific map we want to create (i.e. if we are extending an already existing EM for a new different topology, we should stick to the official grid used by CMS or ATLAS).

- **Validation Checker** : for a internal check, I would like to add a list of SLHA containing exactly the benchmark points that MA5 used for its validation and implementation , i.e. I would like to re-validate the analysis to check if everything in the production (from the choice of parameters to the running of the code) has gone well. For example I would store the cutflows of the benchmark so the user, after the production, can check if she/he can reproduce the MA5(CMS/ATLAS) cutflows.

- **convert.py helper** : the reason why I created a dictionary of analyses implemented in MA5 inside the MA5 output parser are basically two.
  1. we need a mapping between the names of SRs used in the official papers and MA5 implementations;
  2. practical example: convert.py takes as an input for each SR the name of the file containing the data, the topology, the number of events, bkg events and errors. Let's say an analysis has 10 regions; we want 5 different topologies; we need to write by hand 5x10 = 50 times these info in convert.py. But this information is already contained in the script; so I would like to write a txt file that writes exactly the lines needed by convert.py, so that one can easily copy and paste these lines and save a lot of time. I do not want to manipulate the convert.py because this is stored inside the database, and this EM script is independent of SModelS , and also the editing of convert.py requires a certain degree of "human" work and care, and cannot be completely automatised.
- **EM Uncertainties** : MA5 provides uncertainties on the Efficiencies. SModelS currently does not make use of this information, but do we want to keep it in the EM files? Does convert.py (dataPreparation) complain if I add something like " mother daughter EM # uncertainty " i.e. "600 200 0.258 # 0.098 " ?