Simple software for preparation of CTM emission inputs: emPY

Dušan Štefánik

Slovenský hydrometeorologický ústav, Jeséniova 17, 833 15, Bratislava

dusan.stefanik@shmu.sk

Abstrakt

Predložená práca popisuje jednoduchý softvér na prípravu emisií do chemicko-transportného modelu: emPY. Program emPY bol vyvinutý na SHMÚ pre vnútorné účely, avšak je voľne dostupný a stiahnuteľný na stránkach GitHub. Je naprogramovaný v jazyku Python 3. Pozostáva z troch hlavných podprogramov, ktorých účelom je: a) rozmiestniť emisie z emisných inventarizácií do domény chemicko-transportného modelu s definovaným rozlíšením a projekciou, b) vykonať podrobnú speciáciu nemetánových prchavých organických zlúčenín NMVOC a jemných častíc $PM_{2.5}$ na chemické zlúčeniny vyžadované chemicko-transportným modelom, c) na základe časových profilov vytvoriť z ročných emisných vstupov hodinové emisné výstupy v požadovanom formáte netCDF4. Pomocou modelu emPY boli vygenerované emisie pre model CMAQ vo viacerých projektoch. V dvoch rozsiahlejších projektoch bol použitý pri posudzovaní dopadu zníženia emisií na zmenu koncentrácií. Jeden z nich bol projekt v spolupráci so Svetou bankou pod názvom: Reimbursable Advisory Services, výsledky ktorého by mali byť zahrnuté v Stratégii na zlepšenie kvality ovzdušia. Druhým z nich bol medzinárodný projekt pod názvom: Implementation of Air Quality Plan for Malopolska Region Malopolska in a Healthy Atmosphere. Okrem týchto projektov boli výstupy z modelu emPY použité aj pri detailnejšom modelovaní kvality ovzdušia na Slovensku, ktorého výsledkom sú mapy priemerných koncentrácií niektorých znečisťujćich látok v roku 2017 s rozlíšením 1,5 km. Tieto mapy boli taktiež zahrnuté do kompozitných máp FAIRMODE. Výsledné koncentrácie agregované za okresy sú použité v ďalšom projekte MŽP skúmajúcom dopad znečistenia ovzdušia na zdravie obyvateľov pod názvom: Drivers and Health Impacts of the Ambient Air Pollution. Jedným z cieľov prezentovaného článku je snaha o zavedenie dobrej praxe v SHMÚ, aby sa k vyvinutým zdrojovým kódom tvorili stručné používateľské príručky, čím by boli dostupné aj pre iných používateľov.

Kľúčové slová: emisie, časové profily, speciácia, chemicko-transportný model, kvalita ovzdušia

Anotation

A simple software for preparation of emission inputs for chemical-transport (CTM) model was developed. This software consists of three main subprograms that subsequently import various emissions inputs to the model domain, make the speciation of $PM_{2.5}$ and NMVOC pollutants, and disaggregate the annual emissions to the hourly profiles. Final outputs of the program are the CMAQ-ready emission inputs files in standard netCDF format.

Keywords: emissions, time profiles, speciation, grid allocation, chemical-transport model, air quality

1 Introduction - why the software for preparation of CTM emission inputs is needed

The emission inventories are prepared by experts often aggregated on national, regional, or district levels. Usually, they only provide annual sums with very rough speciation - e.g. only values of all non-methane volatile organic compounds NMVOC and fine particles $PM_{2.5}$ are available. However, chemical-transport models require emissions: a) in specific gridded domain, b) speciated according to the chemistry mechanism used in the model (e.g. instead NMVOC it requires emissions in more verbose form like isoprenes, ethane etc.) c) with hourly time variation d) in model-ready format.

The presented software emPY fully solves the steps b), c), d). The step a) is partially solved, the emPY allocates emissions in specific gridded domain specified by arbitrary projection and resolution, but does not provide so called top-down spatial dis-aggregation. Top-down approach means that emissions from the large scale (national level) are dis-aggregated to higher resolutions (to the specific urban areas, roads etc.) using suitable geographical proxy data. This top-down dis-aggregationis currently done by several institutions like TNO or EMEP and are provided to European countries for modelling purposes.

2 Installation of the emPY software

The emPY model is based on the Python 3 programming language. Basic knowledge of this popular, effective and easy to use programming language is required.

Following the steps bellow, you will be able to use the emPY in the proper way:

1. Download and install the Anaconda distribution of Python https://www.anaconda.com/distribution/ .
2. Make a new conda environment https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html by: conda create –name myenv, where myenv is arbitrary name for your new environment in which emPY software will be running
3. Install geopandas library to the myenv environment by: conda install -c conda-forge geopandas
4. Install pytz, netCDF4 libraries
5. test if all required libraries are properly installed by typing in the ipython prompt:
   import geopandas
   import netCDF4
   import numpy
   import pandas
   import shapely
   import pyproj
   import pytz
6. Download the software from the GitHub page https://github.com/dusssaaan/SHMU/emPY/

3 Description and running of the emPY software

The emPY software was developed at SHMU for the preparation of the emission inputs for the CMAQ chemical-transport model. It is fully written in Python programming language and is structured in easily-extensible way - it means that user can easily add some new functionality. It consists of the three main subprograms and some preprocessors. The software was inspired by SMOKE [1] and FUME [2] emission processors. Although above mentioned processors are more universal and robust they were not appropriate for purposes of SHMU . Presented model emPY is just suited for the CMAQ model [3, 4] and for purposes of SHMU, but is free to use

and available in the GitHub site. It does not produce biogenic emissions since this is done in CMAQ internally. The schematic description of the emPY software is shown in Figure 1. The file structure of the software is shown in Appendix A, where the file structure of the output data is also shown. Adhering to this structure is very important in order to avoid possible confusions which can appear in processing of large number of the emission inventories.

The description of the individual subprograms and way of their usage are presented in the following subsections.

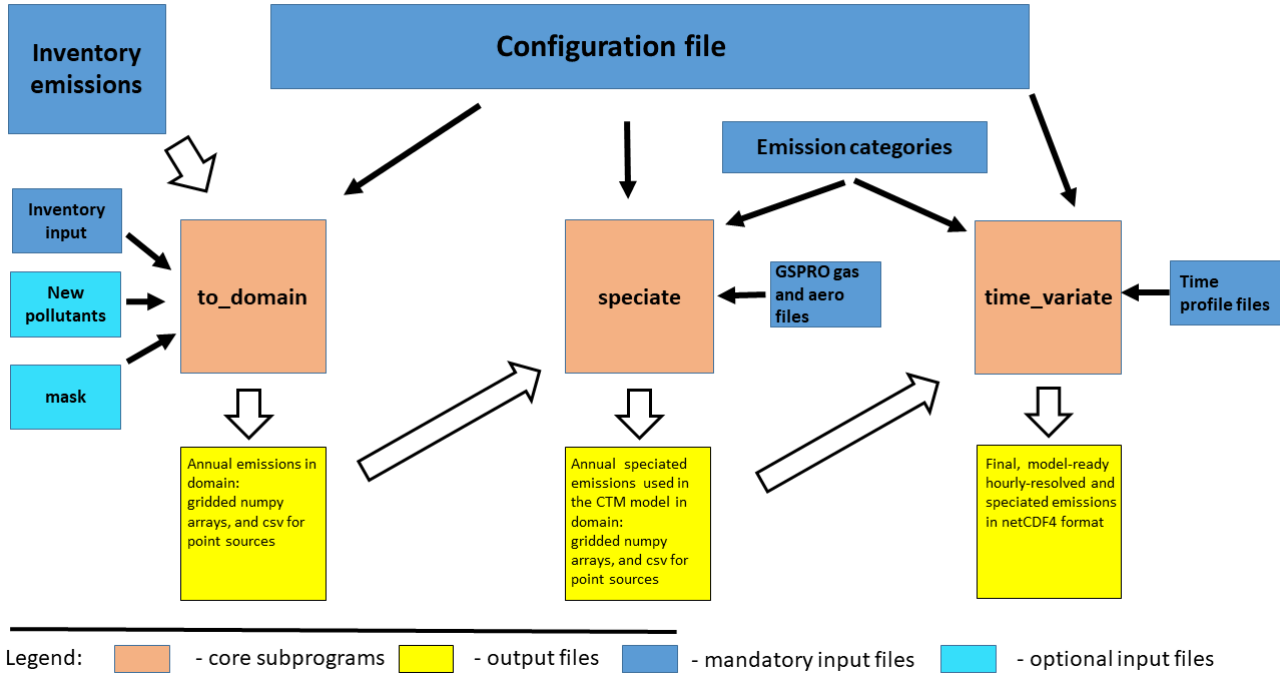

Figure 1: Schema of the emPY software

3.1 to_domain program

Main purpose of the to_domain program is to convert emission inventories of the area, line, and point sources to the gridded domain with projection and resolution of the CTM model. It also translates the specific inventory category numbers and pollutant names from given emission inventories to unified numbers and names used in the model.

The required input files for to_domain program except the proper emission inventory are emPY_config_file.py and inventory_input.py. In the emPY_config_file.py there are general configurations like a definition of projection and grid parameters of the final domain, input and outputs directories, and paths to the input files. In order to import the inventory emissions to the emPY model in a properly way, the inventory_input.py file is required. It contains Python dictionary d. For each emission inventory which user wants to include in the model, the specific item of dictionary needs to be created as follows:

```
d['SVK_cat_02_point']={
'source_type':'P',
'type_file':'csv',
'input_file':'/data/users/emPY/cat2_point_NEIS.csv',
'sep':';',
```

'encoding':'utf-8',
'one_cat':220,
'x':'lon',
'y':'lat',
'def_heights':'zero',
'ID':'id',
'EPSG':4326,
'def_emis':{'tzl_m': 'PM', 'pm10_m': 'PM10', 'pm2_5_m': 'PM2_5', 'so2_m': 'SO2', 'nox_m': 'NOX'},
'units':'t/year', }

From the dictionary item defined above, the emPY knows that the inventory SVK_cat_02_point is a point source inventory which is saved in the file /data/users/emPY/cat2_point_NEIS.csv. Possible keys in d[inventory name] are different for area, line and point sources and are explained and summarised in Tables 1,2, and 3. As explained in Appendix A, do not use the symbol '-' in the inventory name.

The optional files are mask files. One can define a mask for point sources as a shape file with one polygon geometry from which they want to mask out data for a given inventory. For the area sources, a mask in form of a numpy array with values ranging from 0 to 1 is required. This mask can be created within emPY preprocessor tool mask_out.

As one can see from Appendix A, outputs of the to_domain program are numpy arrays for the area and line sources for each internal emission category and pollutant name. For the point sources, outputs are in form of csv files.

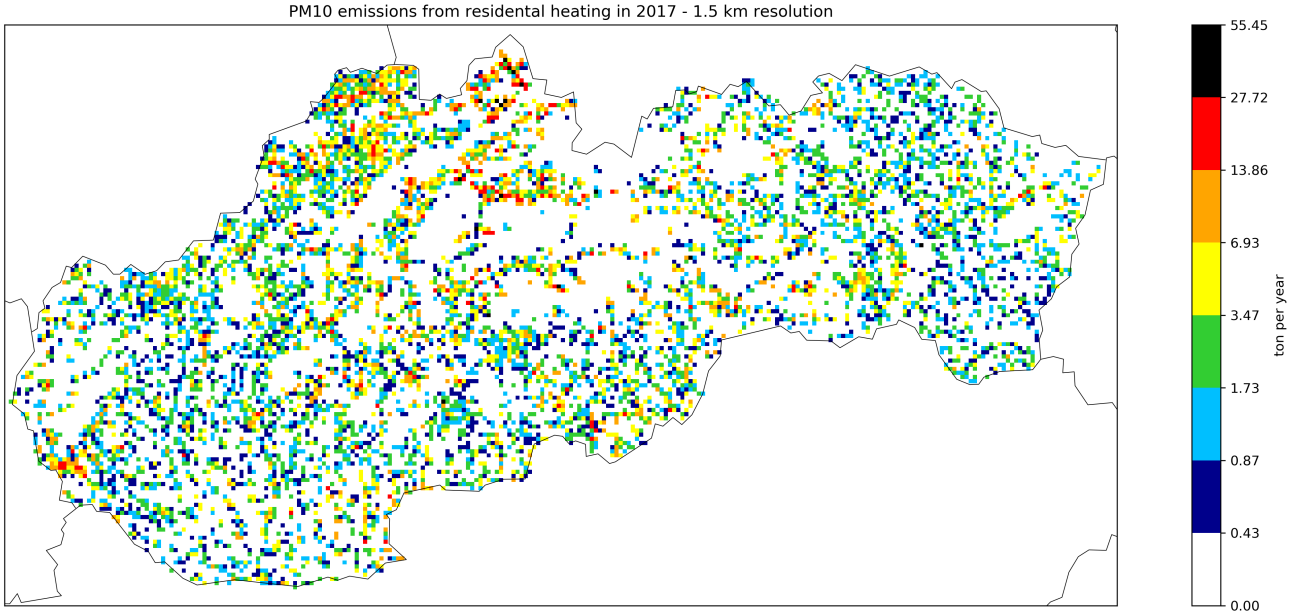

Figure 2: Example of the output from the to_domain program

3.1.1 Area sources

Inventory emissions for the area sources can by in three formats a) in a shape-file with various polygons with specific emissions b) in a csv format, regularly gridded in projection defined by EPSG c) in a combination of a csv file, in which the emissions are specified, and shape file in which the geometries are specified. The possible

keys in dictionary item in inventory_input file for the area sources are in Table 1.

key	values	case in use
source_type	'A'	mandatory
type	csv or csv+shape or shape	mandatory
input_file	string path to the file	if type = csv or csv+shape
sep	separation sign in the input_file, like ',' or ';' etc.	if type = csv or csv+shape
encoding	encoding of input_file like 'utf-8' or other	if type = csv or csv+shape
shape_file	string path to the shape file	if type = shape or csv+shape
cat_internal	string name of columns with inventory number	optional
emission_inventory	dictionary: conversion to internal cat numbers	if cat_internal is presented
one_cat	internal cat number of input file	if cat_internal is not presented
source_id	string name of id columns in input_file	if type = csv+shape
shape_id	string name of id columns in shape_file	if type = csv+shape
new_pollutants	strings with equations how to calculate some new pollutants†	optional
def_emis	dictionary: conversion to internal emission names	mandatory
x	string name of x coordinate columns	if type = csv
y	string name of y coordinate columns	if type = csv
grid_dx	float number of x size of the grid	if type = csv
grid_dy	float number of x size of the grid	if type = csv
EPSG	int number of EPSG projection	if type = csv
units	'mg/year', 'g/year','kg/year', 't/year', 'kilot/year'	mandatory
mask_out	True (if mask_out is not present the default is False) ††	optional

Table 1: Keys in inventory_input file for the area sources

† The equations need to be separated with comma, the most left equation is applied first, then second etc., for example 'NO2= 0.05*NOX, NO= 0.652*NOX-0.652*NO2, BZN=0.016*NMVOC, NMBVOC=NMVOC-BZN'.
†† If mask_out=True the mask array defined in the path mask_area in emPY_config_file.py will be applied

3.1.2 Line sources

Line sources can be defined in the same way as the point or area sources. In case they are defined by shape-file with line geometries, the emPy needs to treat them separately. The possible keys in a dictionary item in the inventory_input file for the line sources are listed in Table 2. Only shape files with one emission category is now allowed in the emPY.

key	values	case in use
source_type	'L'	mandatory
shape_file	string path to the shape file	mandatory
one_cat	internal cat number of input file	mandatory
new_pollutants	strings with equations how to calculate some new pollutants†	optional
def_emis	dictionary: conversion to internal emission names	mandatory
units	'mg/year', 'g/year','kg/year', 't/year', 'kilot/year'	mandatory

Table 2: Keys in inventory_input file for the line sources

3.1.3 Point sources

Inventory emissions for the point sources can be in a shape-file or a csv format. Possible keys and values for the point sources dictionary item are in Table 2 .

If the key 'def_heights' is set to 'zero', point sources are treated as surface area sources and no stack information are required by the emPY model. Output of the to_domain script for point sources with 'def_heights' different from 'zero' are csv files in a separate directory (see the file structure of the output data in Appendix A). If the 'def_heights' is set to False, the function apply_stack_parameters in the module convert_to_empy_names.py is applied and default parameters defined in emPY_config_file.py are given to the stacks according to the parameter dictionary defined in this function. For the explanation, see comments in the convert_to_empy_names.py file.

key	values	case in use
source_type	'P'	mandatory
type	csv or shape	mandatory
input_file	string path to the file	if type = csv
sep	separation sign in the input_file, like ',' or ';' etc.	if type = csv
encoding	encoding of input_file like 'utf-8' or other	if type = csv
shape_file	string path to the shape file	if type = shape
cat_internal	string name of columns with inventory number	optional
emission_inventory	dictionary: conversion to internal cat numbers	if cat_internal is presented
one_cat	internal cat number of input file	if cat_internal is not presented
new_pollutants	strings with equations how to calculate some new pollutants†	optional
def_emis	dictionary: conversion to internal emission names	mandatory
x	string name of x coordinate columns	if type = csv
y	string name of y coordinate columns	if type = csv
def_heights	True or False or 'zero'	mandatory
ID	string name of columns with ID stack number	if def_heights = True
height	string name of columns with ID height	if def_heights = True
diameter	string name of columns with ID diameter	if def_heights = True
temperature	string name of columns with ID plum temperature	if def_heights = True
velocity	string name of columns with ID plum velocity	if def_heights = True
EPSG	int number of EPSG projection	if type = csv
units	'mg/year', 'g/year','kg/year', 't/year', 'kilot/year'	mandatory
mask_out	True (if mask_out is not present the default is False) *	optional

Table 3: Keys in inventory_input file for the point sources

* If mask_out = True the mask shape file defined in the path mask_point in emPY_config_file.py will be applied.

3.2 speciate program

Main purpose of the speciate program is to speciate non-methane volatile organic compounds (NMVOC) and fine particles $PM_{2.5}$ to the more detailed species required by the model. It requires the following files: emPY_config_file.py, emission_categories.csv, and spec_file.csv which has similar structure as the GSPRO files for aerosols and gases. In emPY_config_file.py, user sets the input and output data file path and specifies the list of the emission inventories which he wants to include to the model outputs. In emission_categories.csv file, user needs to assign a speciation number profile according which the speciaton will be provided, for the each emission category which he defined in the inventory_input.py file, and he wants to include to the model. In the

spec_file.csv for each speciation number, there is a line with name of a unspeciated species, name of a species after the speciation, coef1, coef2 and coef3. If the speciation number is 0 in the spec_file.csv it means that the speciation will be done for every category. The speciation is done as:

emission of the specied pollutant = emission of the unspecied pollutant* coef1/coef2.

The specied emissions are then in the kmols for gasses or tons for particles. The GSPRO files are same as in the SMOKE model [5].

Beside speciation the speciate program also group all point sources from the output of the to_domain program to one csv file and creates CMAQ-ready netCDF4 file with stack parameters.

3.3 time_variate program

Main purpose of the time_variate program is to prepare daily CMAQ-ready emission files with hourly time steps. It requires next files: emPY_config_file.py, emission_categories.csv, and time_profile files. In config_file.py user sets the input and output data file path, date range of the simulation, time_zone of the domain, and names and units of the pollutants which can be write in the final netCDF file. In emission_categories.csv file user needs to assign for the each emission category which he wants to include it to the model, the time_profile number according which the time variation will be provided. The time variation is then done by the default time profiles defined in the files tv_map.csv and tv_values.csv, or by specific time_factors defined in tv_series.csv

4 Run the emPY software

Following next steps one can run the emPY sofware

- cp -r case_run case_run_orig
- Be careful, the program read configuration from case_run
- adjust the emPY_config_file.py, inventory_input.py, emissions_categories.csv, or possibly all other files in case_run directory
- activate your proper python environment as conda activate my_env
- cd ${HOME emPY directory}/to_domain
  ./run_to_domain.py > ../case_run/log_to_domain
  after successfully running of the program you should obtain output data in numpy array files and csv with the structure as in the Appendix A . This files represent yearly emissions in the domain in tons.
- cd ${HOME emPY directory}/speciate
  ./run_speciate.py > ../case_run/log_speciate
  after successfully running of the program you should obtain output data in numpy array files which contains yearly speciated emissions. For the point sources you obtain one csv files with yearly speciated emissions and one CMAQ-ready netCDF4 file with stacks parameters.
- cd ${HOME emPY directory}/time_variate
  ./run_time_variate.py > ../case_run/log_time_variate
  after successfully running of the program you should obtain for each day in simulation 2 kinds of the output data in CMAQ-ready netCDF4 files one for the area surface emissions and second for the elevated point sources.

- After finishing your simulation cp -r case_run case_run_{name_of_archived_simulation}. When you will start the new simulation you need to adjust script in the case_run folder.

If you want to view intermediate data produce during the run of the emPY software, Python offers a lot of libraries for great visualisation. We offer simple tool npyview in tools directory, which can be used as follows:

npyview 2D_numpy_array.npy

you just need to add path to your .bashrc file for example as:

alias npyview='path_to_npyview_tool/npyview.py'

5 Projects in which the emPY software was used

Meantime the emPY software was used in two large project in which the impact of emission reduction scenarios on the concentrations was assessed using the chemical-transport model. One was the Reimbursable Advisory Services (RAS) project by World Bank. The results of this project should be included in the Air Protection Strategy of Slovak Republic. The second project in which emPY was used is international LIFE-IP Integrated Project "Implementation of Air Quality Plan for Malopolska Region Malopolska in a healthy atmosphere". In this project 101 emission inventories are used as inputs to emPY and it is a most complex case in which the emPY was tested. Besides this two project the emPY software was also used as the emission input for the CMAQ detailed 1.5 km resolution simulation, from which the air quality maps for the year 2017 was done. These maps was also included in the Fairmode EU Composite Maps exercise. The emPY software was also used as an input in the calculation of district concentrations of specific air pollutants in Slovakia which is now used in the project: Drivers and health impacts of the ambient air pollution.

6 Conclusions

The software for preparation of CTM emission inputs - emPY is presented. emPY is an open-source product developed at SHMU for internal purposes, but is available free on GitHub for any user. It prepares CMAQ-ready emission files. The software is written in Python 3 programming language. The emPY software has already been used for the preparation of the emission inputs for CMAQ model for several projects One of the aims of the presented paper is the effort to introduce good practice in the developing code at SHMU which also includes preparation of documentation and user guides. Without documentation is not possible to share the code with other potential users.

Acknowledgements

I would like to express gratitude to Nina Benešová and Ondrej Vlček from the Czech Hydrometeorological Institute, Peter Huszar and Michal Belda from the Charles University for their detailed explanation of the general features of emission processors. Also, I would like to thank to my colleges from the OMKO SHMÚ department for great working atmosphere.

References

[1] Baek, B. H., Seppanen, C., and Houyoux, M.: SMOKE v2.6 user's manual, http://www. smoke-model.org/version2.6/,

[2] Benešová et al.,(2018): New open source emission processor for air quality models. In Sokhi, R., Tiwari, P. R., Gállego, M. J., Craviotto Arnau, J. M., Castells Guiu, C. Singh, V. (eds) Proceedings of Abstracts 11th International Conference on Air Quality Science and Application. DOI: 10.18745/PB.19829. (pp. 27). Published by University of Hertfordshire. Paper presented at Air Quality 2018 conference, Barcelona, 12-16 March.

[3] U.S. EPA. EPA's Report on the Environment (Roe) (2008 Final Report). U.S. Environmental Protection Agency, Washington, D.C., EPA/600/R-07/045F (NTIS PB2008-112484), 2008

[4] Štefánik, D., 2017: Air quality modeling using the CMAQ model, Zborník príspevkov 18. konferencia mladých meteorológov a klimatológov, SHMÚ Bratislava, 22.-24.11.2017 ISBN 978-80-88907-95-4

[5] https://www.cmascenter.org/smoke/documentation/2.1/html/ch08s05s02.html#tbl_input_gspro

A Appendix: File structure of the emPY program and its outputs

The file structure of the emPY software is following:

```
emPY/
|
+ - - case_run/
|       |
|       + - - emPY_config_file.py
|       + - - inventory_input.py
|       + - - emission_categories.csv
|       + - - spec_file.csv
|       + - - tv_map_em.csv
|       + - - tv_series.csv
|       + - - tv_values.csv
|
+ - - to_domain/
|       |
|       + - - run_to_domain.py
+       | - - src_to_domain/
|               |
|               + - - area_to_domain.py
|               + - - convert_to_empy_names.py
|               + - - point_to_domain_zero.py
|               + - - point_to_domain.py
|       |
+ - - speciate/
|       |
|       + - - run_speciate.py
+       | - - src_speciate/
|               |
|               + - - group_point_sources.py
|       |
+ - - time_variate/
|       |
|       + - - run_time_variate.py
+       | - - src_time_variate/
|               |
|               + - - gridded_emision_time.py
|               + - - point_emision_time.py
|               + - - time_matrix.py
|       |
+ - - tools/
|       |
|       + - - npyview/
|               |
|               + - - npyview.py
|       |
|       + - - mask_out/
|               |
```

```
|                    + − − run_mask_out.py
|                    | − − src_mask_out/
|                                |
|                                + − − mask_out.py
```

The file structure of the output data should to be following:

```
${HOME emPY directory}/data/outputs/
|
+ − − outputs-to_domain-{name of case_run}/
|       |
+       | − − {inventory_name_1}/
|                   |
|                   + − − {id_cat_1}-{internal_pollutant_name_1}-{inventory_name_1}.npy
|                   + − − {id_cat_1}-{internal_pollutant_name_2}-{inventory_name_1}.npy
|                   + − − {id_cat_2}-{internal_pollutant_name_1}-{inventory_name_1}.npy
|                   .
|                   .
|                   .
+       | − − {inventory_name_2}/
|                   |
|                   .
|                   .
|                   .
+       .
|       .
|       .
+       | − − point_sources/
|       |
|                   + − − {point_inventory_name_1}.csv
|                   + − − {point_inventory_name_2}.csv
|                   .
|                   .
|                   .
|
+ − − outputs-speciate-{name of case_run}/
|       |
|       + − −{id_cat_1}-{speciated_pollutant_name_1}.npy
|       + − −{id_cat_1}-{speciated_pollutant_name_2}.npy
|       + − −{id_cat_2}-{speciated_pollutant_name_1}.npy
|                   .
|                   .
|                   .
+       | − − point_sources/
|       |
|                   + − − speciate_points.csv
|                   + − − STACK_PARAM.nc
|       |
+ − − outputs-time_variate-{name of case_run}/
|       |
|                   + − −AREA_EMISSIONS-YYYY-MM-DD.nc
|                   + − − POINT_EMISSIONS-YYYY-MM-DD.nc
```

Note that the symbol '-' is important in the data structure of the output files since it serves as a separator symbol. From this reason it is forbidden to use symbol '-' in inventory and pollutant names.