EpiC Tool v1.0
Submodule Documentation
October 14, 2010

Bruno Gonçalves

Contents

1 Single Population Epidemic Model (SPEM)

1.1 SPEMShell.py

`SPEMShell.py` is Python script that is in charge of running the simulations. The wrapper takes care of running the executable(s) to produce the output files.

When calling it, a single parameter is passed: the name of the directory in which the input files will be found (a relative subpath starting from a pre-defined base path).

The directory (let's call it SIMDIR) should contain three files called:

- `req.cfg`
- `simul.in`
- `simul.mdl`

The `req.cfg` file should contain two lines:

RUNS: (integer) 1 for single/live run, > 1 for multiple runs

OUTVAL: (string) a list of compartment labels, as defined in the model, separated by ";" (semicolon)

OUTVAL defines the output quantity to be displayed on the visualization client application and is interpreted to mean the sum of all the transitions in to the compartments specified.

When `SPEMShell.py` is invoked the SIMDIR will contain only the 3 conf files (req.cfg, simul.in, simul.mdl) and an empty directory called `output`

At the end of the simulation (after the last step/travel files have been saved inside the "frames" subdir) the wrapper should create an empty file inside SIMDIR called `run.end` used, especially for multiple runs, to confirm that all the computation is finished.

The data files of each computation step, and all aggregates generated can be found in the `<SIMDIR>/output` directory, in order to be accessible for further analysis/visualization.

1.2 Simulation Output

Multiple runs

For each simulation successfully ended there will be a directory called:

`<ROOT_DIRECTORY>/output/`

where <ROOT_DIRECTORY> is the simulation directory that was passed to SPEMShell.py. Inside the output_data directory there will be, for each of the N runs, three files called, respectively, SPEM.<r>.out, SPEM.<r>.err and SPEM.<r>.sec where <r> is a progressive number identifying the run, ranging from 0 to $N-1$ when N is the number of runs (the value set in the Simulation Requirements).

SPEM.<r>.out These files contain the complete description of the instantaneous populations in each compartment at every time step.

The first column corresponds to the time step, followed one column for each compartment defined in the model. The sequence of compartments is documented in the first line of the file that starts with a comment char "#" followed by the word "time". No blank lines are allowed and every line (including the first one) has the same number of columns, where this number varies from model to model. All entries are listed, even ones composed of all 0's.

SPEM.<r>.err These files contain complete information about all the transitions that occurred during the run. Only non-zero values are listed.

Each line contains four columns, a transition ID, the time step, the number 0, and the number of individuals that followed that transition in that specific time step. The transition ID is consistent between runs and it listed in the respective GLEaM.<r>.out.dat (notice the lack of the trailing ".gz") in an intuitive notation. For example, for a simple SIR model, this might be:

0 (S) – (I) (I) 0.7
1 (I) -> (R) 0.36

where 0.36 and 0.7 are the respective rates. The ID number indicates the order by which it was parsed from the model input file and are used simply as a short hand to refer to a given transition.The rate values listed here are the final numerical values used after all the appropriate substitutions and calculations have been done.

SPEM.<r>.sec These files contain a subset of the information contained in the previous ones. Again, only non-zero values are listed. Here we list the total number of secondary cases occurring in each time step. This corresponds to a simple aggregation of the corresponding transitions listed in the respective SPEM.<r>.err. These files have a two column format, time step and number of secondary cases.

comp.<c>.dat For each compartment, <c> defined in the model, a file is generated containing the average population and 95% confidence intervals for each time step.

outVal.dat The outVal.dat file contains information relevant to the outVal compartments specified in the simulation requirements. Each line, has 7 columns, timestep, the total average number of transitions into an outVal compartment followed by the 95% confidence interval. The final 3 columns are reserved for the cumulative values.

sum.<a>.<t>.dat Finally, a second set of files can also be found in the output directory. The respective files names are of the form:

- sum.basins.<t>.dat
- step-<date>.1.sum.dat

where <t> the time step and <date> is the corresponding date.

sum files contain three columns, the number 0, the number of transitions in to the compartments listed in the requirements file as output variables, and it's cumulative value. By aggregating each of these files, we obtain the step files containing one line of 7 columns, in the same format as the outVal.dat file, with the difference that the first column is always 0. Here all values are listed (even zeros) and each file will have N lines, where N is the number of runs. These files are an intermediate step in producing the final output shown in the output visualization, but the user might find them useful for other purposes as well.

Single Run

In the case of a single run, all the files retain their formats, except in what concerns CI as these are no longer defined.

2 Exact Model Integration (ExactEM)

In this case, the model, defined as above, is converted into the respective partial differential equations that are then integrated numerically using an adaptive Runge-Kutta procedure.

2.1 ExactShell.py

ExactShell.py is a Python script similar in function to SPEMShell.py. The requirements are the same with a few small differences. Since in this case we are dealing with a deterministic, integration, the RUNS parameter is ignored. Also, at this point, OUTVAL is not supported as it is not clear

how to implement it in the current integration procedure. Otherwise, the format of the output files is similar to the one defined for `SPEMShell.py` in the case of single runs.

3 Epidemic Models on a Network (NetEM)

For the network case, we consider each individual to be a node on a network interacting with its nearest neighbors. In this formulation, since we are considering single individuals, all values defined in the model are considered to be probabilities instead of rates as above. The exact value of R_0 depends on the properties of the network.

3.1 `NetEMShell.py`

`NetEMShell.py` works similarly to the scripts defined above. OUTVAL is correctly defined, and, in addition to producing the same files as `SPEMShell.py` one extra file can be produced.

`NetEM.*.state` If the "frames" parameter is set to a directory name in the req.cfg file, a file `NetEM.*.state` will be generated for each run containing the status of every node on the network at each time step. This file can be used to generate an animation showing the disease progression on the network. While this option is also valid in the case of multiple runs, it is probably only useful for single runs as it is not clear how to aggregate it over various instances.