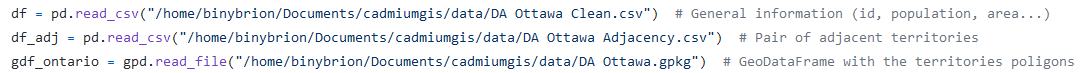
Contents

[Overview 2](#_Toc50131282)

[Example 3](#_Toc50131283)

# 

# Overview

To use the input generator:  
  
Modify the scenario.json to contain the format of the default values. So for example, given  
the following input file format:  
  
 \*\* Point A \*\*  
{  
 "scenario": {  
 "default\_delay": "inertial",  
 "default\_cell\_type": "zhong",  
 "default\_state": {  
 "population": 1,  
 "age\_group\_proportions": [0.5, 0.5],  
 "susceptible": [1, 1, 1],  
 "fatalities": [0, 0, 0],  
 "infected": [  
 [0, 0, 0,0],  
 [0, 0, 0],  
 ],  
 "recovered": [  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 ],  
 "disobedient": 0.25,  
 "hospital\_capacity": 0.2,  
 "fatality\_modifier": 1.5  
 },  
 "default\_vicinity": {  
 "correlation": 1,  
 "infection\_correction\_factors": {}  
 },  
 "default\_config": {  
 "zhong": {  
 "precision": 100000000,  
 "virulence\_rates": [  
 [0.15, 0.15, 0.15],  
 [0.15, 0.15, 0.15],  
 ],  
 "recovery\_rates":  
 [  
 [0.07, 0.07, 0.07],  
 [0.07, 0.07, 0.07],  
 ],  
 "mobility\_rates": [  
 [0.6, 0.6, 0.6],  
 [0.6, 0.6, 0.6],  
 ],  
 "fatality\_rates": [  
 [0.005, 0.005, 0.005,],  
 [0.005, 0.005, 0.005]  
 ],  
 "SIIRS\_model": true  
 }  
 }  
 }  
   
 \*\* Point B \*\*  
 ,  
  
 "cells": [ define some cells... ]  
}  
  
Copy everything between Point A and Point B into the scenario.json file, and add an ending  
brace at the end of the file.  
  
Next, open the generate\_cadmium\_json.py file. Go to section # In[16].  
Change the 'df', 'df\_adj' and 'gdf\_ontario' variables to point to the required files. For example:  
  
  
Finally, go to section # In[24].

Change the state variable to contain the format each state should have. It should match format of the default state in the scenario.json file (though with optionally different values). The addition to the cell's neighbourhood, done with the 'adj\_full[str(row["dauid"])]["neighborhood"].append ...' should also  
be changed to match the required neighbourhood / vicinity structure in the scenario.json file.  
  
In the # In[26] section, make sure to change the location of where the output file is written.  
  
Finally, run the program: python generate\_cadmium\_json.py (Note: python3 may be required)

# Example

For example, suppose that a scenario in the Analysis/Input folder is to be run. Let’s say that the scenario (a cut down version of) absolute\_threshold\_02\_01 is to be run, which is located in the AbsoluteLockdown\_SIIRS\_NoDisobedient.py file.

Notice how the JSON is stored in a Python file (allows those scenarios to be run from a python script if needed, if the cells defined in those scenarios are adequate, like in the Analysis document) ; thus a little bit of extra work needs to be done.

To use this scenario using cells from real world data, the following would be done. Copy the content between Point A and Point B as mentioned before. A screenshot of this is done below.



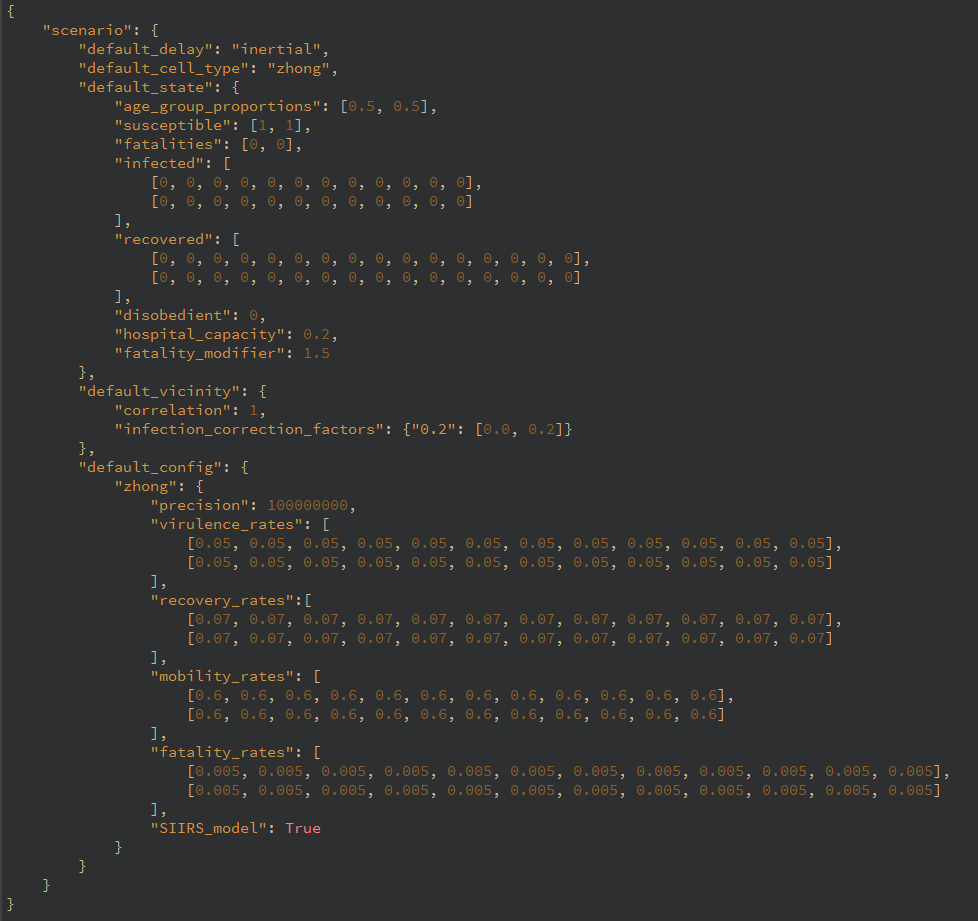
Now paste this into scenario.json, adding an additional brace at the end, removing the variable name (“absolute\_threshold\_02\_01” in this case) and changing the

“SIIRS\_model”: True

to

“SIIRS\_model”: ***true***

(change is in bold and italicized; this is done because in JSON boolean keywords start with a lowercase, not a upper case. Additionally, commas with no item afterwards may have to be removed, such as after the end of the last subarray in an array.

The scenario.json will now look like:

Now in the generate\_cadmium\_json.py file, in section #In[24], the state and vicinity variable must have the same format as the default\_state in the scenario.json file, like so:

The state in this section is the initial state every cell extracted from the real world data will have. While the format has to be the same as what is in the scenario.json, the values can be different. For example, notice how the “population” variable was changed to be whatever is the population of a real world cell, rather than a hardcoded 1.

Next, the neighborhood format needs to be changed to match what was in the scenario.json:

Here, the lockdown threshold was changed to start at 0.3, not 0.2. This was done to show the values can differ from the scenario.json in the vicinity structure as well.

Lastly change where the file is written to:

