**NatalCovid19**

**Input parameters**

*population* Input population parameters: age population range, age population distribution, disease probability of death, ICU, asymptomatic and symptomatic by age, etc. Layers of system: home, random, religion, school, transport and work.

*epidemy* Input epidemy parameters: virus contamination, time of incubation, discharge and hospital and number initial of infected agents.

*sim* Input sim parameters: seed and days of simulation.

**\_\_main\_\_.py**

Is a file on directory: “./covidsimulation” and is the main code of the system. Verify the input parameters, run the main simulation, create and save the output data.

Functions

*buildJSONfromArgs ()*

Read the input parameters and create a single .json objectfrom passed arguments. If any output directory was not passed use the default: "./analysis/sim\_results/".

*validateJSON (params)*

Validate the minimum input information to run the simulation according to the minimums parameters (population, epidemy and sim).

*output (output\_file, data)*

Create output data file and save on the specified directory (if passed on input) or default output directory (./analysis/sim\_results/). If the output already contains data then the new data will be appended to the data file.

*main ()*

The main function of the file and the system, calls the other three functions above cited and the *run* functions from: "./simulation/covid\_simulation.py". This *run* function will execute the simulation of the agents' interactions over the days.

**agent.py**

File on directory: “./covidsimulation/simulation/agent.py”. It is the class to simulate the agent on the system.

**class Agent**

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data type** | **Description** |
| age\_group | Int | Age of agent. |
| health\_state | Int | Actual health state. |
| health\_outcome | Int | Outcome health state. |
| days\_exposed | Int | Days exposed of agent. |
| days\_with\_symptoms | Int | Days with agent symptoms. |
| incubation\_time | int | Days of incubation. |

Methods

*cycleDisease (self, sim)*

Verify the current health state, days exposed and update it accordingly to the disease cycle. If infected increase days of exposure; if with symptoms, increase days with symptoms.

*\_cycleNoAction (self, sim)*

Not defined yet.

*\_cycleIncubated (self, sim)*

If contaminated and half the incubation period has passed, becomes *asymptomatic*.

*\_cycleAsymptomatic (self, sim)*

If the *health\_outcome* is equal an asymptomatic and after a period of time (7 days) becomes immune; if not, advance the disease and the health state become *symptomaticLight.*

*\_cycleSymptomaticLight (self, sim)*

If the *health\_outcome* is equal *asymptomaticLight* after a period of time (14 days) becomes immune; else, advance the disease and the health state become *symptomaticMedium*

*\_cycleSymptomaticMedium (self, sim)*

If the *health\_outcome* is equal *symptomaticMedium* and a period of time (14 days) has passed, becomes immune. If *health\_outcome* is different of *symptomaticMedium*, advance the disease and go to the hospital state. If days with symptoms pass (7 days) the *health\_outcome* becomes hospital state or ICU state.

*\_cycleHospital (self, sim)*

If in the hospital and alive after a period of time (28 days) becomes immune.

*\_cycleICU (self, sim)*

After a period of time (22 days), if *health\_outcome* equal ICU state, becomes hospital state and hereafter recovery, if outcome is equal dead the state becomes *dead*.

**data\_collector.py**

File on directory: “./covidsimulation/simulation/data\_collector.py”. Collect agent health state, store day by day and save at the end of the simulation.

Functions

*countHealthStates (population)*

Count the number of people in each health state and store on a list.

**class DataCollector**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| health\_states | List | Number of people each health states. |

*collect (self, population)* Stores health state population for the current day on *health\_states* list.

*serialise (self)* Converts the object representation to a dictionary object to save on output.

**health\_states.py**

File on directory: “./covidsimulation /common/health\_states.py”. A file with the health states used on the system. The nine states are:

* Susceptible - No exposure to the disease
* Immune - Had the disease and recovered
* Dead - Had the disease and died
* Incubated - Exposed to the disease but not yet exhibiting symptoms
* Asymptomatic – Have the disease and do not show symptoms
* Symptomatic light – Have the disease and shows light symptoms
* Symptomatic medium - Have the disease and shows medium symptoms
* Hospital – Have the disease and have to go the hospital
* ICU – Have the disease and have to go to ICU

**network.py**

File on directory: “./covidsimulation/simulation/network.py”. As this system is based on a multi-layer structure, six layers were implemented: home, random, religion, school, transport and work. Each layer represents the interaction between the agents.

**class Network**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| layers | list <NetworkLayer> | List of layers. |

Methods

*calculateInteractions (self, population, p\_contamination)*

Chose and make the interaction between agents and calculates the probability of contamination in each layer it belongs to.

**layer.py**

File on directory: “./covidsimulation/simulation/layers/layer.py”.

Functions

*chooseUniform (a, size)*Choose agents to interact with based on network criteria.

*isExposed (p\_interaction, p\_contamination)*

Define the health state of those susceptible people exposed to virus base on the probability of interaction and contamination.

**class NetworkLayer**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| **nodes** | **Dictionary** |  |

Methods

*ChooseContacts (self, agent)*

Return a list of the agents that have been contacted this time step.

*getPInteraction (self, agent)*

Return the probability for close interaction with this agent.

*\_calculateInteractionProbability (self, exposed\_hours, group\_factor)*

Calculates the probability of interaction (agent to agent).

*\_calculateInteractionProbabilityAndGroupFactor (self, exposed\_hours, number\_of\_contacts, number\_of\_connections)*

Calculates the probability of interaction (group to group).

*\_createConnectionsUniform (self, population, filtered\_indexes, min\_group\_size, max\_group\_size)*

Creates a new connection between groups of agents.

*\_createConnections (self, population, filtered\_indexes, min\_group\_size, max\_group\_size, size\_distribution\_func)*

Creates small world with new connection between groups of agents.

**home.py**

File on directory: “./covidsimulation/simulation/layers/home.py”. *Home* layer is the interactions between the agents who live in the same home. Create a connection among agents on the same home based on family size and calculate the probability of interaction.

**class Home**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| p\_interaction | Float | Probability of interaction among agents who lives together. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**random.py**

File on directory: “./covidsimulation/simulation/layers/random.py”. *Random* layer is the interaction between agents choose randomly. Create a connection between two random agents and calculates the interaction probability.

**class Random**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| p\_interaction | Float | Probability of interaction. |
| average\_num\_contacts | Int | Average number of contacts of each agent. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**religion.py**

File on directory: “./covidsimulation/simulation/layers/religion.py”. *Religion* layer is the interaction between agents who frequents religious temples. Create a small world with connections among the agents frequents and calculates the interaction probability.

**class Religion**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| exposed\_hours\_per\_week | Int | Number of hours exposed per week. |
| average\_num\_contacts | Int | Average number of contacts of each agent. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**school.py**

File on directory: “./covidsimulation/simulation/layers/school.py”. *School* layer is the interaction between agents who frequents school. Create a small world with connections among the agents who frequents the school with the same age and calculates the interaction probability.

**class Religion**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| exposed\_hours\_per\_week | Int | Number of hours exposed per week. |
| average\_num\_contacts | Int | Average number of contacts of each agent. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**transport.py**

File on directory: “./covidsimulation/simulation/layers/transport.py”. *Transport* layer is the interaction between agents who use same transport to move over the city. Create a small world with connections among the agents who use transport and calculates the interaction probability.

**class Transport**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| exposed\_hours\_per\_week | Int | Number of hours exposed per week. |
| average\_num\_contacts | Int | Average number of contacts of each agent. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**work.py**

File on directory: “./covidsimulation/simulation/layers/work.py”. *Work* layer is the interaction between agents who works. Create a small world with connections among the agents who work and calculate the interaction probability.

**class Work**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| exposed\_hours\_per\_week | Int | Number of hours exposed per week. |
| average\_num\_contacts | Int | Average number of contacts of each agent. |

Methods

*getPInteraction (self, agent)*

Get probability of interaction of agent

*chooseContacts (self, agent)*

Make contact of everyone at home.

**covid\_simulation.py**

File on directory: “./covidsimulation/simulation/covid\_simulation.py”. Create all agents and Run main simulation. Executes interaction of all agents over the days, collect and save all the output data.

Functions

*setNumbaSeed(seed)* Set the initial disease seed.

**class COVIDSimulation**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| seed | Int | Initial disease seed |
| days | Int | Days to simulated |
| p\_contamination | Float | Contamination probability |
| t\_incubation | Int | Incubation time |
| t\_discharge | Int | Discharge time |
| t\_hospital | Int | Hospital time |
| initial\_infected | Int | Initial number of agents infected |
| population\_total | Int | Total of population to simulated |
| population | List <Agent> | Agents who will be simulated |
| network | Network | Network of layers |

Methods

*run (self)*

Run the main simulation with the interaction of all agents over the days, collect and return the data output.

*\_assignHealthOutcomes (self, ages, outcome\_params)*

Define the probability distribution of health outcomes for each age group.

*\_createPopulation (self, ages, outcome\_params, incubations)*

Create all agents used in the simulation.