Pandemia: User Guide

November 28, 2022

Contents

1 Introduction

Pandemia is an individual-based stochastic pandemic simulator. It if able to simulate and visualize the spread of an infectious disease across multiple geographical regions. These regions could represent the countries of the world, or the administrative divisions of a single country. The model is fast and scalable, able to simulate extremely large numbers of individuals, while supporting a wide range of highly adaptable features.

This software can be used by researchers looking to assess the impact of policy in the context of a public health emergency caused by an infectious disease of humans. The disease could, for example, be a respiratory infectious disease spread by a coronavirus or an influenza virus.

The emergence and re-emergence of infectious diseases threatens the health and well-being of people all over the world, and tools such as Pandemia can play a vital role in supporting pandemic preparedness and response.

This user guide presents an introduction to the software and its various components.

- parallel processing - scale factor - random seed

2 World

The Pandemia simulator acts upon a **World**. A **World** consists of **Regions**, with each **Region** consisting of **Agents**, **Locations** and **Activities**. Each agent performs a sequence of activities, and performs these activities in particular locations. A **World** additionally consists of a **Travel Matrix**, representing how many agents travel from each region to each other region each day. Here is the simplest possible **World** structure:

- One **Region**, one **Activity**, one **Location** and any number of **Agents**.

Here is the most complicated:

- Several **Regions**, with each **Region** consisting of several **Activities**, any number of **Locations** and any number of **Agents**, with a **Travel Matrix** describing the mixing between regions.

2.1 Agents

An individual human being is referred to as an **Agent**. Each agent is described by their age, a weekly routine of activities, locations at which the agent might perform these activities, and weights indicating the probabilities of these locations being chosen.

2.2 Locations

A **Location** represents, for example, an area of land or a building, such as a house, restaurant, shop or school classroom. A location is described by its type, and a pair of spatial coordinates.

2.3 Activities

An **Activity** is something an agent does, for example cooking, driving to work, or shopping. Activities are simply labels, with no deeper structure.

2.4 Regions

A **Region** represents, for example, a country or administrative division. A region consists of a set of activities, a set of agents, and a set of locations.

2.5 World Factory

A World is built by a **World Factory**. Different worlds are built either by changing the configuration of a world factory, or by using a different world factory entirely.

2.6 Vectorization

Once a **World** has been built, it is then converted into a **Vector World**. This is done by converting each **Region** into a **Vector Region**. A **Vector Region** is a vectorized version of a **Region**, in which data is formatted as arrays of integers and floats, as opposed to Python lists and dictionaries. This facilitates interface with libraries of functions written in C.

3 Simulator

The model features of a number of components, most of which are optional:

- **Movement**
- **Travel**
- **Health**
- **Hospitalization**
- **Testing and Contact Tracing**
- **Vaccination**
- **Seasonality**

Additional or alternative components are easily added. Pandemia provides a default model for each component. The details of these default models are described in separate documents. The main Pandemia loop looks approximately as follows:

```
for day in clock:
  travel.dynamics(day)
  for region in regions:
    seasonality.dynamics(region, day)
    input.dynamics(region, day)
    for tick in range(ticks_in_day):
      t = (ticks_in_day * day) + tick
      health.dynamics(region, t)
      movement.dynamics(region, t)
      hospitalization.dynamics(region, t)
    testing_and_contact_tracing.dynamics(region, day)
    vaccination.dynamics(region, day)
```

Note that some components update each day, while others update each tick. At the beginning of each day, Pandemia decides who is travelling from each region to each other region, and infects these travellers based on the average infectiousness of their destination region. These travellers are then set aside for the remainder of the day. Pandemia then loops over all the regions, performing independent individual-based simulations on a finer timescale. Since these simulations are independent, the loop over the regions can be parallelized. In this way, Pandemia is both fast and scalable. Even on a laptop computer, using 15 CPUs and 24GB of RAM, it has been able to perform a 100 day simulation, with a 1 hour tick length, of over 100 million individuals, in under 1.5 hours.

The **Input** component allows the user to specify a **Policy**, consisting of interventions. Possible interventions include the following:

- **Lockdown**
- **Border Closure**
- **Vaccination**
- **Testing and Contact Tracing**
- **Quarantine**
- **Face Masks**

Additional or alternative interventions are easily added. **Reporters** collect output data for visualization and analysis.

The **World** object, having been built, can be saved and thus it is not necessary to complete the build phase each time a simulation is run. The **World** object is fed into the simulator.

3.1 Clock

3.2 World

3.3 Components

3.4 Simulator Factory

3.5 Reporters

4 Movement

The Pandemia model acts on a **World**, with each **World** consisting of **Regions**, with each **Region** consisting of **Individuals**, **Locations** and **Activities**. Each individual performs a sequence of activities and performs these activities at particular locations. During the building of the **World**, each individual is assigned a weekly routine, starting on a Sunday, consisting of a sequence of activities.

For example, suppose in a given region there are three activities, **Home**, **Work** and **School**, and that tick length is set equal to 8 hours, meaning that there are 3 ticks of the simulation clock per day. Then a possible weekly routine could be:

[**Home**, **Home**, **Home**, **Home**, **Work**, **Home**, **Home**, **Work**, **Home**, **Home**, **Work**, **Home**, **Home**, **Work**, **Home**, **Home**, **Work**, **Home**, **Home**, **Home**, **Home**]

representing a typical working week, or alternatively

[**Home**, **Home**, **Home**, **Home**, **School**, **Home**, **Home**, **School**, **Home**, **Home**, **School**, **Home**, **Home**, **School**, **Home**, **Home**, **School**, **Home**, **Home**, **Home**, **Home**]

representing a typical school week.

During the building of the **World**, each individual is assigned a weighted set of locations for each activity. These are the locations at which an individual can perform each activity. During the simulation, whenever an individual switches from one activity to another, the individual randomly selects a new location from the set of allowed locations, according to the weights.

For example, suppose in a given region there are two activities, **Home** and **Other**. Then, a given individual might be assigned a single location for the activity **Home**, with weight 1.0, and several other locations for the activity **Other**, with weights chosen according to the distance of each location from the home, so that locations further from home are less likely to be visited, and so on.

Upon changing activity, individuals may also put on or take off a face mask, depending on the activity and the current policy on face masks.

If the change of location is prohibited by policy interventions, then the individual is directed to their home location. In particular, each individual must for the default movement model be assigned a home location.

5 Travel

6 Health

Epidemic models typically represent health using discrete states. In a typical compartmental model, the population is partitioned into subsets, labelled *Susceptible*, *Infected*, *Recovered* and *Dead*. These compartments can then be subdivided, and new compartments added, to produce increasingly complicated models.

This approach runs into difficulties once partial immunity is introduced, since the compartmental label *Recovered* then becomes ambiguous. It runs into further difficulties once multiple strains of the pathogen are introduced, since then the label *Susceptible* becomes ambiguous.

Pandemia therefore dispenses with the compartmental framework altogether, taking an entirely new approach to modelling individual health. In Pandemia, the health of an individual is described by five attributes:

- **Strain**
- **Disease**
- **Infectiousness**
- **Immunity (outer layer)**
- **Immunity (inner layers)**

The immune system is represented using layers, with the outer layer determining whether or not an infection is blocked, and the inner layers determining the outcome for infections that are not blocked. A key innovation of Pandemia is that, for each individual, these five attributes are stored as *functions*.

Variables

The value of an attribute at time `t` is given by evaluating the corresponding function at time `t`. During a simulation, the following variables store the values of these functions at the current time:

- `current_strain`
- `current_disease`
- `current_infectiousness`
- `current_sigma_immunity_failure`
- `current_rho_immunity_failure`

The prefix `sigma_` refers to the outer layer, while the prefix `rho_` refers to the inner layers.

current_strain

For individual n, the variable

current_strain[n]

indicates whether or not individual n is infected, and if so with which strain. The variable is an integer, taking values in the range

{-1, 0, 1, 2, ..., S - 1}

where S denotes the number of strains. If the individual is not infected, then

current_strain[n] = -1.

We assume that individuals can only be infected with one strain at a time.

current_disease

For individual n, the variable

current_disease[n]

indicates the extent to which individual n is diseased. The variable is a float, taking values in the range [0,1]. If the individual has no disease, then

current_disease[n] = 0.

If the individual is dead, then

current_disease[n] = 1.

Values close to 1 represent severe illness, values close to 0 represent mild illness. Values above a threshold represent symptomatic illness, values below the threshold represent asymptomatic illness.

current_infectiousness

For individual n, the variable

current_infectiousness[n]

indicates the extent to which individual n is infectious. The variable is a float, taking non-negative values. If the individual is infected but not infectious, then

current_infectiousness[n] = 0.

If the individual is infected, then increasing the value of this variable increases the probability that the individual transmits strain current_strain[n] to other individuals.

current_sigma_immunity_failure

For individual `n` and strain `s`, the variable

```
current_sigma_immunity_failure[n][s]
```

represents the probability that the immune system of individual `n` *fails* to prevent an infection when confronted with strain `s`. Being a probability, this variable is therefore a float taking values in the range `[0,1]`. Since this variable stores a failure probability, values closer to `0` represent higher levels of protection.

current_rho_immunity_failure

If an infection has occurred, then sigma immunity has failed and the pathogen has made it past the outer layer of defence. It now confronts a number of internal layers. Each layer has a probability of failing to stop the pathogen, with the outcome of the infection getting progressively worse the deeper the pathogen penetrates. This binary tree structure allows the user to parametrize, for example, the efficacy of a vaccine against symptomatic illness, severe illness and death. If the pathogen passes through all but the last internal layer, which is impenetrable, then the individual experiences the worst possible outcome.

For example, suppose the model is configured in such a way that there are two internal layers. Then, for a given individual, infected with a given strain, there are two possible outcomes. If the first internal layer fails to stop the pathogen, the outcome will be the latter of the two outcomes. If the first internal layer is successful, then the outcome will be the former of the two outcomes. These two outcomes could, for example, be configured to represent mild and severe illness.

More generally, suppose the model is configured in such a way that there are `R` internal layers (and therefore `R` possible outcomes). Then, for individual `n` and strain `s`, the variable

```
current_rho_immunity_failure[n][s]
```

gives an array of probabilities, of length `R`, corresponding to the *failure* probabilities of each layer. If a pathogen makes it past layer `r`, then it moves on to face layer `r+1`, else the individual gets outcome `r`. Since the last layer is impenetrable, the final entry in this array is always set equal to `0`, meaning that the last layer never fails to stop the pathogen. This procedure is completed immediately after infection.

While sigma immunity determines whether or not an infection occurs, with rho immunity determining the outcome of that infection, to understand the outcomes themselves we must discuss the `health_presets`.

Presets and Updates

Pandemia determines the default response to an infection for each individual, for each strain, in advance of the simulation. In particular, if an individual has been assigned the

preset response `p`, from the set of possible preset responses `health_presets`, then the object `p[r]` contains parameters which determine updates for each of the five health attributes described above, corresponding to outcome `r`.

For example, suppose that our model features two strains and two internal layers. Then the object `p[r]` may look as follows:

```
rho_immunity_failure:
  [[[[-1, 5, 30], [[1.0, 0.0], [0.25, 0.0], [1.0, 0.0]]],
    [[-1, 5, 30], [[1.0, 0.0], [0.50, 0.0], [1.0, 0.0]]]],
   [[[-1, 6, 30], [[1.0, 0.0], [0.50, 0.0], [1.0, 0.0]]],
    [[-1, 6, 30], [[1.0, 0.0], [0.25, 0.0], [1.0, 0.0]]]]]
sigma_immunity_failure:
  [[[[-1, 5, 30], [1.0, 0.25, 1.0]],
    [[-1, 5, 30], [1.0, 0.50, 1.0]]],
   [[[-1, 6, 30], [1.0, 0.50, 1.0]],
    [[-1, 6, 30], [1.0, 0.25, 1.0]]]]
infectiousness:
  [[[-1, 0, 3, 5], [0.0, 0.0025, 0.0075, 0.0]],
   [[-1, 0, 3, 6], [0.0, 0.0055, 0.0085, 0.0]]]
disease:
  [[[-1, 0, 5], [0.0, 0.2, 0.0]],
   [[-1, 0, 6], [0.0, 0.3, 0.0]]]
strain:
  [[[-1, 0, 5], [-1, 0, -1]],
   [[-1, 0, 6], [-1, 1, -1]]]
```

The numbers in these arrays encode step functions. For example, the pair of arrays `[[-1,0,5], [0.0,0.2,0.0]]` encodes the step function f given by

$$f(t) = \begin{cases} 0.0 \text{ for } t < 0 \\ 0.2 \text{ for } 5 > t \geq 0 \\ 0.0 \text{ for } t \geq 5 \end{cases} . \tag{1}$$

Suppose in the example that individual `n` has just been infected with strain `0`, and that the infection has resulted in outcome `r`, with `p[r]` as above. Then the variable `current_strain[n]` will take the value `0` for the next `5` days, at which time it returns to the value `-1`, indicating that the individual is no longer infected. The variable `current_disease[n]` will take the value `0.2` for the next `5` days, at which time it returns to the value `0.0`, indicating that the individual has recovered. The variable `current_infectiousness[n]` will take the value `0.0025` for the next `3` days, followed by `0.0075` for `2` days, after which time it returns to the value `0.0`, indicating that the individual is no longer infectious.

Updates to the sigma and rho immunity variables are more complicated. The immunity variables are updated via the operation of function multiplication. Recall that the immunity functions store probabilities of failure, so the product of such functions give the probabilities of failure for overlapping immune responses, assuming independence.

Suppose in the above example that individual `n` has no immunity against either strain prior to infection. Assume that the infection with strain `0` occurred at time `t`. Then, after updating their immunity functions, the component of their sigma immunity function corresponding to strain `0` will be given by the step function

```
[-1, t + 5, t + 30], [1.0, 0.25, 1.0],
```

while the component corresponding to strain `1` will be given by

```
[-1, t + 5, t + 30], [1.0, 0.5, 1.0].
```

In particular, for 25 days following the end of their infection, the probability that their immune system fails to protect against another infection by strain `0` is improved from `1.0` to `0.25`, after which it returns to `1.0`, representing a loss of immunity. The probability that their immune system fails to protect against an infection by strain `0` is improved from `1.0` to `0.5`, after which it returns to `1.0`, representing a loss of cross immunity.

Suppose now that at time `t + 10` individual `n` is again infected by strain `0`. Then between times `t + 10` and `t + 15`, the three variables describing their current strain, disease and infectiousness will be updated as before. But the component of their sigma immunity function corresponding to strain `0` will now be subject to appropriate multiplication, after which it will be given by the step function

```
[-1, t + 5, t + 15, t + 30, t + 40],

[1.0, 0.25, 0.0625, 0.25, 1.0].
```

In particular, between times `t + 15` and `t + 30` there are overlapping immunity responses, so that in order for a third infection to occur by strain `0`, the pathogen must overcome the immune response generated by the first infection *and* the immune response generated by the second infection. Our assumption is that these events are independent, hence the multiplication of probabilities.

Updates to rho immunity are similar, except that these functions are now array-valued, these arrays corresponding to the failure probabilities for each internal layer, so the functions must be multiplied component-wise.

For a preset `p` and outcome `r`, while the components of `p[r]` corresponding to strain, disease and infectiousness are vectors, each containing precisely `S` functions, where `S` is the number of strains, the components corresponding to rho and sigma immunity are matrices, containing precisely $\texttt{S} \times \texttt{S}$ functions, the additional dimension accounting for possibility of cross-immunity, as in the above example.

Transmission

For a location l in region i, we define:

$$p_l := \left(1 - \prod_{m \in l} (1 - f_m)\right)$$

where $m \in l$ means all individuals m currently in location l, with

$$f_m := \omega_m \nu_m \mu(i) \lambda(l) \beta(s_m).$$

Here

- ω_m is the current face mask multiplier associated to individual m, which takes the value 1 if they are not wearing a face mask, and some number smaller that 1 if they are;
- ν_m is the current infectiousness of individual m;
- μ is a multiplier depending on the region i, reflecting for example seasonal changes in transmission that act on the regional level;
- λ is a multiplier depending on the location l, reflecting the fact that some types of location might be less conducive to transmission than others;
- β is a control coefficient depending on the strain s_m that individual m is currently infected with, if any.

For each susceptible individual n in location l, with current face mask multiplier ω_n, the probability that they are exposed at this time is then assumed to be $\omega_n p_l$. If such an individual is exposed, then to determine which strain they are exposed to, we assume the expression

$$\frac{\sum_{\{m \in l : s_m = s\}} f_m}{\sum_{\{m \in l\}} f_m}$$

gives the probability that they are exposed to strain s. Given this exposure, the probability that they are actually infected with strain s is then σ_{ns}, the current sigma immunity of individual n against strain s. The outcome of this infection is then determined by the rho immunity of the individual, according the procedure outlined in the previous subsections.

SIR Rescaling

If the option `sir_rescaling` is set to `True`, then the transmission probabilities are rescaled in such a way that approximates the homogeneous mixing of standard compartmental models. Moreover, this option allows for the activation of a mixing matrix, that can be used to describe mixing between population subgroups, for example age groups.

For example, suppose we have only one region, one location, one strain and no face masks. Suppose that $\nu_m = 1$ if m is infected, with $\nu_m = 0$ otherwise. Denote by N_a the number of people in group a and by h the step size. Then the rescaling factor for the transmission probabilities becomes h/N_a. In particular, denoting by m_{ab} the mixing between groups a and b and by I_b the number of currently infected individuals in age

group b, for an individual n in group $a(n)$, we have:

$$\begin{aligned} p_n &= 1 - \prod_b \left(1 - \frac{h\beta m_{a(n)b}}{N_b}\right)^{I_b} \\ &\approx 1 - \prod_{b \in A} \exp\left(-\frac{h\beta m_{a(n)b} I_b}{N_b}\right) \\ &= 1 - \exp\left(-\sum_b \frac{h\beta m_{a(n)b} I_b}{N_b}\right) \\ &= 1 - \exp\left(-h\beta (MI)_{a(n)}\right) \end{aligned}$$

where $M_{ab} := m_{ab}/N_b$ is the normalized mixing matrix. This is consistent with a standard compartmental model, since the expected number of new infections in age group a at time t then satisfies the approximation

$$\begin{aligned} \mathbb{E}\left[\frac{S_a(t+h) - S_a(t)}{h}\right] &\approx -S_a(t)\left(\frac{1 - \exp\left(-h\beta (MI(t))_a\right)}{h}\right) \\ &\approx -\beta S_a(t)(MI(t))_a. \end{aligned}$$

In particular, with only one population subgroup we have $M = 1/N$, therefore recovering the first equation

$$\frac{d}{dt}S(t) = -\beta \frac{S(t)I(t)}{N}$$

of the SIR model. With exponentially distributed recovery times, and no reinfection, we recover the remaining equations, and therefore arrive at a stochastic approximation of the SIR model.

With the option `sir_rescaling` set to `True`, the default health model is therefore a stochastic individual-based generalization of the SIR model. On the other hand, with the option `sir_rescaling` set to `False`, the transmission probabilities are not divided by the number of individuals in each location, meaning that adding susceptible individuals to the location of an infected individual does not dilute the infectiousness of that individual, as it does under the SIR rescaling. For example, if Alice and Bob are riding a bus, and Bob is infectious, then under the SIR rescaling the probability that Alice is infected by Bob decreases if more susceptible people get on the bus. This may not be realistic, so by default the rescaling is set to `False`.

7 Hospitalization

8 Testing and Contact Tracing

9 Vaccination

10 Seasonality

11 Input

12 Optimization