Ornstein Uhlenbeck Pandemic Model

Peter Cotton

May 8, 2020

Abstract

We describe a simple Orstein-Uhlenbeck motion model implemented in the Python pandemic package available on PyPi and Github.

1 Simulation model

Individuals are particle on the plane with Poisson health states. The simulation comprises:

1. A city simulation phase in which home and work locations are generated,
2. A movement phase where particles follow random walks on the plane to and from work according to a regime switching Ornstein-Uhlenbeck process

During the movement phase, infection is driven by near collisions (particles with the same approximate positions). The simulation is governed by parameters broken down into motion, geometry and health categories. The full set of system parameters are given in Table 2.1.

1.1 Health model

Particles are assigned a state and these transition with fixed rates. Particles may be vulnerable (i.e. susceptible), infected (but not symptomatic), symptomatic, positive, recovered or deceased. See Table 2.1 There are nine non-zero entries in the transition matrix out of a possible 36. See pandemic/health.py

1. Infected may progress to symptomatic (is)
2. Infected may be randomly tested and proceed to positive (ip)

3. Symptmatic may be tested and proceed to positive (sp)
4. Infected may recover (ir)
5. Positive may recover (pr)
6. Symptomatic may recover (sr)
7. Infected may die (id)
8. Positive may die (pd)
9. Symptomatic may die (sd)

The corresponding rates are given by concatenation of the state abbreviate and looking up the health parameter. For example the Poisson rate for positive people dying is given by params['health']['pd']

1.1.1 Compliance

Sick people stay home, which is to say that their attractors are set to their home locations.

1.2 Motion model

Particles are assigned two attractors on the plane (some have only one). These represent home and work (where work could represent school or some other location visited daily). The parameter params['geometry'][c'] governs the percentage of people who have two attractors versus one.

1.2.1 OU Walks

Motion of particles in the plane is intended to approximate OU walks.

$$dx_t = \kappa(\theta(y_t) - x_t) + \Sigma dW_t$$

where $\kappa > 0$ is a fixed parameter denoted by k in the code. Here x_t and $\theta \in \mathcal{R}^2$ and Σ is diagonal. We are using the standard notation for OU processes. Here θ are termed attractors in the code (work or home). The variable y_t represents a deterministic regime that takes one of two values. The regime switches twice a day.

See pandemic/motion.py

1.2.2 Collisions

At a fixed number of times per day (parameter 't') a set of approximate locations of infected people is computed by means of a geohash (interleaving of the digits of the binary representation). The parameter params['geometry']['p'] governs the precision of this geohash and thus the proximity that constitutes a collision.

Collisions lead to transmission of infection with a probability params['health']['vi']. See movement.py and the function that computes newly exposed people.

1.3 Home and work location generation

The city simulation vaguely resembles a Chinese restaurant process. Given one location $X \in \mathcal{R}^2$ for a home, a new home location is chosen by

$$x'. = (1+s)x + ex|x| + r\eta$$

where $\eta \sim N(0,1)$, $|x|$ is the distance from the origin and s, r are fixed parameters (the spawl and sprawl quadratic term mentioned above). This spawl process is computed for home and work locations separately (with homes using a linear factor of $4r$ versus r for work). In addition, multiple centers are employed by default.[1]

2 Usage

See the examples folder. Typical usage would be as follows:

```
from pandemic.example_parameters import SMALL_CITY
from pandemic.simulation import simulate
import matplotlib.pyplot as plt
simulate(params=SMALL_CITY,plt=plt)
```

2.1 Callbacks

At present the typical way for a user to extend the simulation or add metrics is via a callback. See pandemic/surrogate.py for an example of a callback that logs running metrics to a database.

[1] Users of the code can of course substitute their own home and work locations as they wish.

State	Abbreviation	Meaning
Vulnerable	v	Susceptible population
Infected	i	Infected but not symptomatic
Symptomatic	s	Infected and symptomatic
Positive	p	Confirmed positive
Recovered	r	Recovered
Deceased	d	Deceased

Category	Abbreviation	Description
Motion	t	Number of time steps used in motion simulation per day
Motion	k	Mean reversion parameter κ (pull towards home or work)
Motion	w	Corresponds to σ in the Brownian walk
Geometry	n	Population count
Geometry	i	Initial infected count
Geometry	r	Radius
Geometry	b	Bound on the size of the world
Geometry	h	Mean household size
Geometry	c	Communing fraction
Geometry	s	Sprawl
Geometry	c	Sprawl quadratic term
Health	vi	Mean transition rate from vulnerable (v) to infected (i)
Health	is	Mean transition rate from infected (i) to symptomatic (s)
...		
Health	pr	Mean transition rate from positive (p) to recovered (r)

Table 1: Full set of system parameters. Note that other health parameters are similar as they are merely labels for entries in a transition matrix as described in Section **??**.