Anyone can publish on Medium per our Policies, but we don’t fact-check every story. For more info about the coronavirus, see cdc.gov.

Modelling the coronavirus epidemic in a city with Python

Are cities prepared for epidemics?

Gevorg Yeghikyan Follow
Feb 4 · 10 min read ★

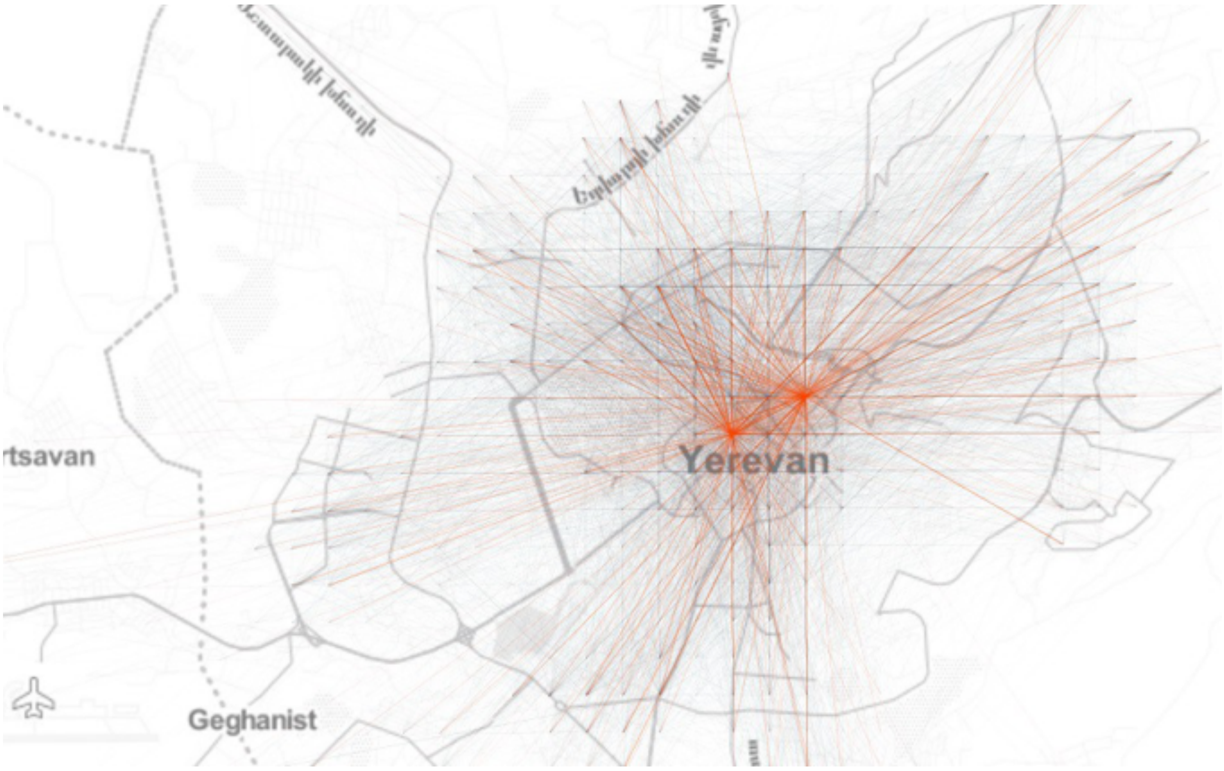
The recent 2019-nCoV Wuhan coronavirus outbreak in China has sent shocks through financial markets and entire economies, and has duly triggered panic among the general population around the world. On 30 January 2020, 2019-nCoV was even designated a global health emergency by the World Health Organization (WHO). At the time of this writing, no specific treatment verified by medical research standards has yet been discovered. Moreover, some key epidemiological metrics such as the basic reproduction number (the average number of people infected by an ill individual) are still unknown. In our times of unprecedented global connectedness and mobility, such epidemics are a major threat on a global scale due to small world network effects. One could conjecture that conditional on a global catastrophic event (loosely defined as > 100mln casualties) happening in 2020, the most likely cause would be precisely some pandemic — not a nuclear disaster, not climate catastrophe, etc. This is further aggravated by worldwide rapid urbanisation, with our densely populated dynamic cities turning into propagation nodes in the disease diffusion network, thus becoming extremely vulnerable and fragile.

In this post, **we will discuss what can happen when an epidemic strikes a city, what measures should immediately be taken, and what implications this has for urban planning, policy making, and management**. We will take the city of Yerevan as our case study and will mathematically model and simulate the spread of the coronavirus in the city, looking at how urban mobility patterns affect the spread of the disease.

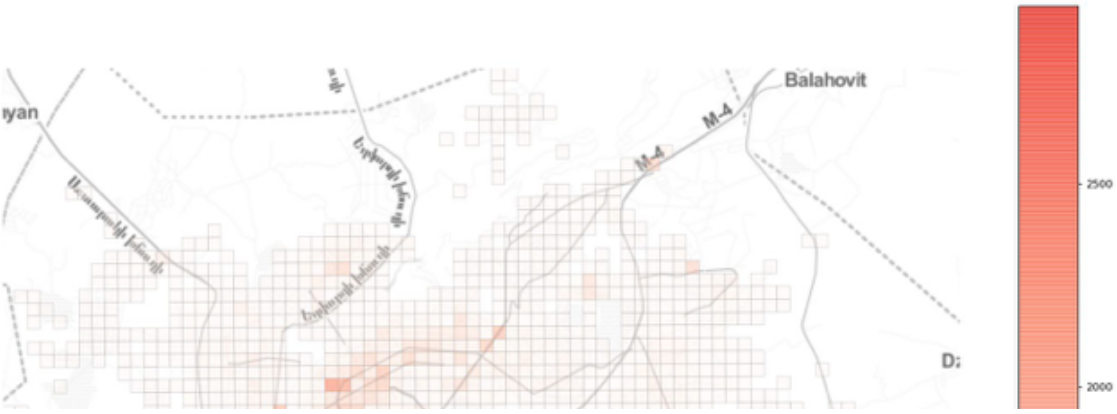
Urban mobility

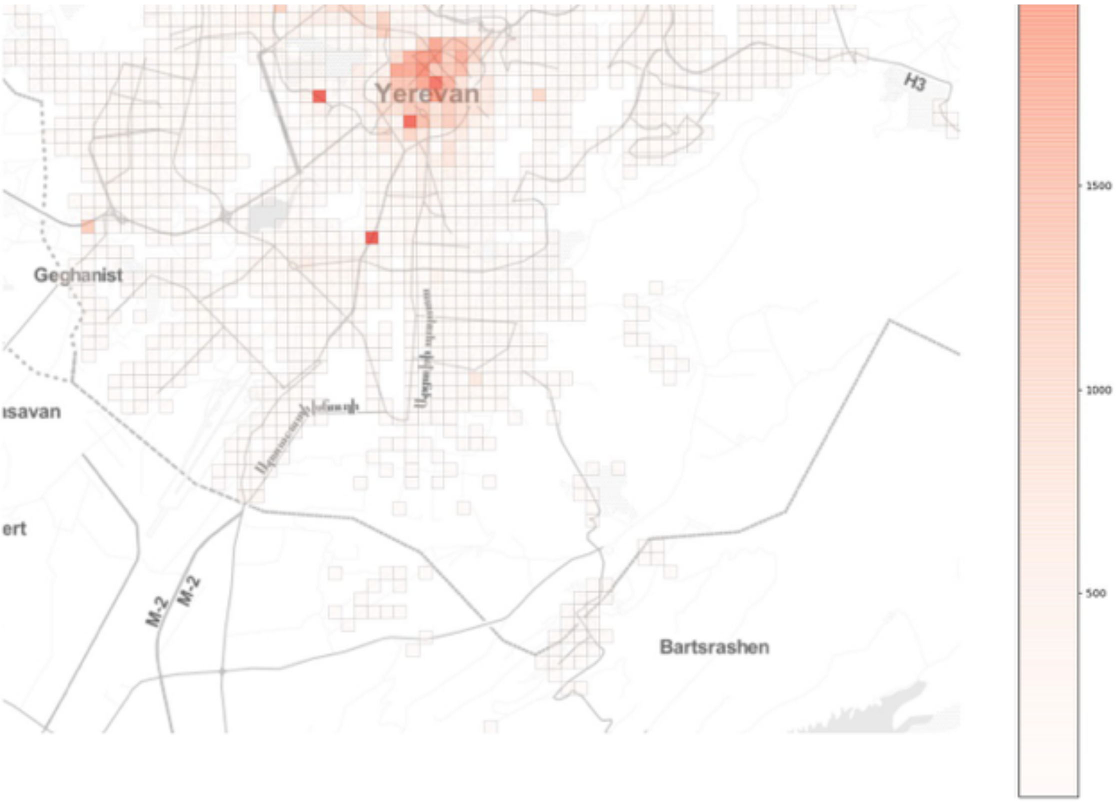
Effective, efficient, and sustainable urban mobility is of crucial importance for the functioning of modern cities. It has been shown to directly affect livability and economic output (GDP) of cities. However, **in the event of an epidemic, it will add fuel to the fire**, amplifyig and propagating the disease spread.

So let's begin by looking at the network of aggregated origin-destination (*OD*) flows on a uniform Cartesian grid in Yerevan to get an idea about the spatial structure of mobility patterns in the city:



Further, if we look at the total inflow to the grid cells, we see a more or less monocentric spatial organisation with some cells with high daily inflow located off the center:





Now, imagine that an epidemic breaks out at a random location in the city. **How will it spread? What can be done to contain it?**

Modelling the epidemic

To answer these questions, we will build a simple compartmental model to simulate the spread of infectious disease in the city. As an epidemic breaks out, its transmission dynamics varies significantly, depending on the geographical locations of the initial infection and its connectivity with the rest of the city. This is one of the most important insights gained from recent, data-driven studies on epidemics in urban populations. However, as we will see further below, the various outcomes call for similar measures to contain the epidemic and to account for such a possibility in planning and managing cities.

Since runnning individual-based epidemic models is challenging, and since our goal is to show general principles of epidemic spread in cities, and not to build a minutely calibrated and accurate epidemic model, we will follow the approach described in this Nature article, modifying the described classical SIR model for our needs.

The model divides the population into three compartments. For each location *i* at time *t*, the three compartments are as follows:

- *Si,t*: the number of individuals not yet infected or susceptible to the disease.
- *Ii,t*: the number of individuals infected with the disease and capable of spreading the disease to those in the susceptible group.
- *Ri,t*: the number of individuals who have been infected and then removed from the infected group, either due to recovery or due to death. Individuals in this group are not capable of contracting the disease again or transmitting the infection to others.

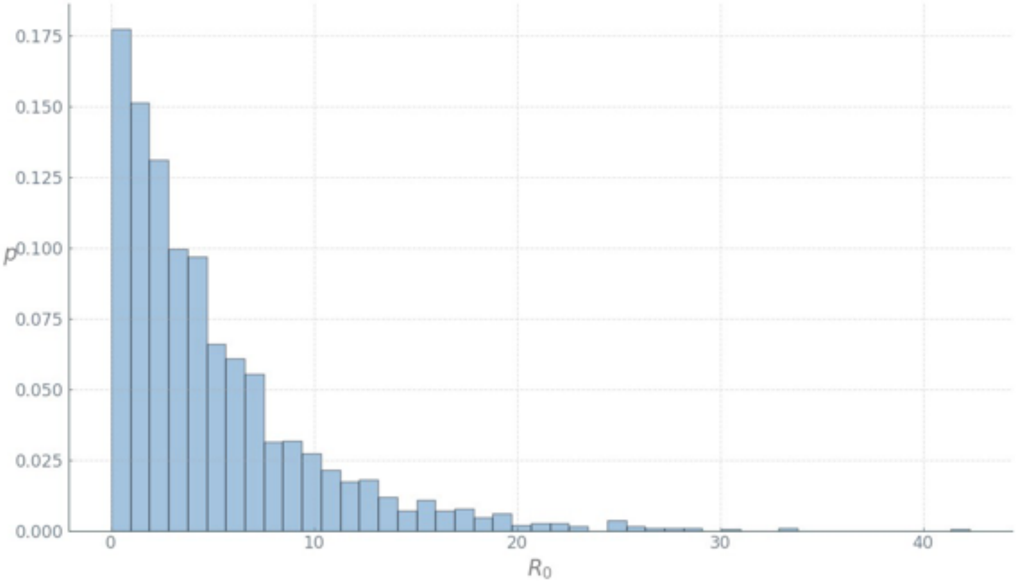
In our simulations, time will be a discrete variable as the state of the system is modelled at a daily basis. In a fully susceptible population at location j at time t, an outbreak happens with probability:

$$h(t,j) = \frac{\beta_t S_{j,t}\left(1 - \exp\left(-\sum_k m^t_{j,k} x_{k,t} y_{j,t}\right)\right)}{1 + \beta_t y_{j,t}},$$

where βt is the transmission rate on day t; mj,k reflects mobility from location k to location j, xk,t and yk,t denote the fraction of the infected and susceptible populations on day t at location k and location j, respectively, given by $xk,t = Ik,t / Nk$ and $yj,t = Sj,t / Nj$, where Nk and Nj are the population sizes at the locations k and j. Then we go ahead and simulate a stochastic process introducing the disease into locations with entirely susceptible populations, with $Ij,t+1$ being a Bernoulli random variable with probability $h(t,j)$.

Once the infections are introduced at random locations, the disease spreads both within those locations and is carried and transmitted in other locations by travelling individuals. **This is where the urban mobility patterns characterised by the *OD* flow matrix play a crucial role**.

Further, to formalise how the disease is transmitted by an infected person, we need the *basic reproduction number, R0*. It is defined as $R0 = \beta t / \gamma$ where γ is the recovery rate, and can be thought of as the expected number of secondary infections after an infected individual comes into contact with a susceptible population. At the time of this writing, the basic reproduction number for the Wuhan coronavirus has been estimated to be between 1.4 and 4. Let's take the worst case and assume it's 4. However, we should note that it's actually a random variable and the reported number is but the *expected* number. To make things a bit more interesting, we will run our simulations with different *R0* at each location, drawn from a good candidate distribution, *Gamma*, with mean 4:



We can now proceed to the model dynamics:

$$S_{j,t+1} = S_{j,t} - \frac{\beta_{j,t} S_{j,t} I_{j,t}}{N_j} - \frac{\alpha S_{j,t} \sum_k m^t_{j,k} x_{k,t} \beta_{k,t}}{N_j + \sum_k m^t_{j,k}}$$

$$
\begin{aligned}
I_{j,t+1} &= I_{j,t} + \frac{\beta_{j,t} S_{j,t} I_{j,t}}{N_j} + \frac{\alpha S_{j,t} \sum_k m^t_{j,k} x_{k,t} \beta_{k,t}}{N_j + \sum_k m^t_{j,k}} - \gamma I_{j,t} \\
R_{j,t+1} &= R_{j,t} + \gamma I_{j,t},
\end{aligned}
$$

where *βk,t* is the (random) transmission rate at location *k* on day *t*, and *α* is a coefficient denoting the modal share or the intensity of public transport vs. private car travel modes in the city.

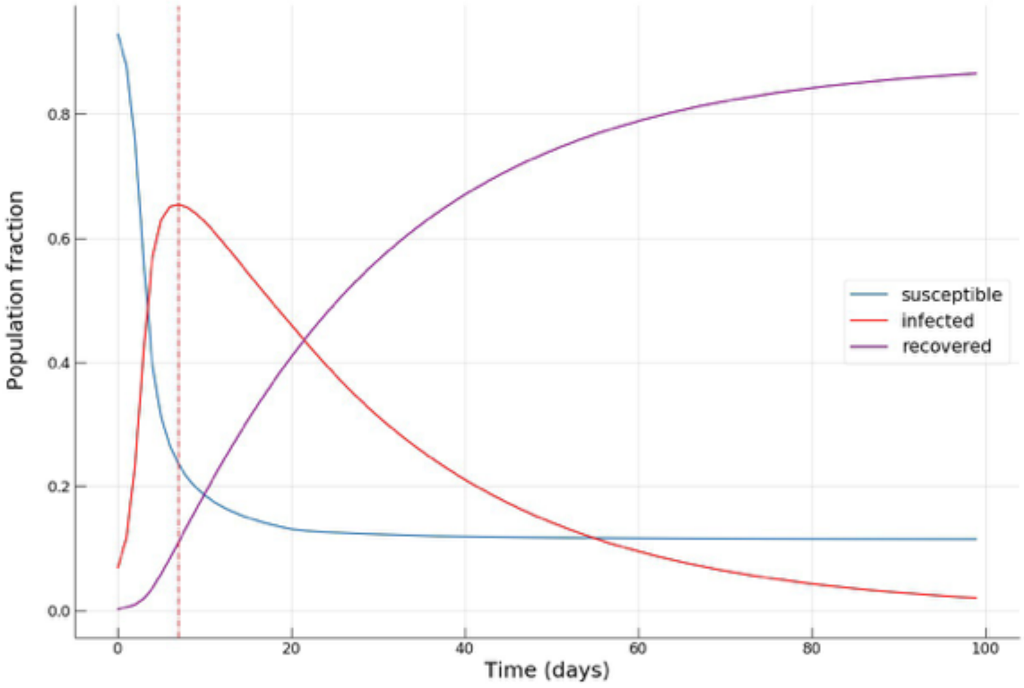
The model dynamics described in the above equations are very simple: on day *t+1* at location *j*, we need to subtract from the susceptible population *Sj,t* the fraction of people infected within location *j* (the second term in the first equation) as well as the fraction of infected people that have arrived from other locations in the city, weighted by their respective transmission rates *βk,t* (the third term in the first equation). Since the total population *Nj = Sj + Ij + Rj*, we need to move the subtracted portion to the infected group, while also moving those recovered to *Rj,t+1* (second and third equations).

Simulation setup

For this analysis, we will use the aggregated *OD* flow matrix of a typical day obtained from GPS data provided by local ride sharing company **gg** as a proxy for the mobility patterns in Yerevan city. Next, we need the population counts in each *250×250m* grid cell, which we approximate by proportionally scaling the extracted flow counts so that the total inflows in different locations sum up to approximately half of Yerevan's population of 1.1 million. This is actually a bold assumption, but since varying this portion yielded very similar results, we will stick to it.

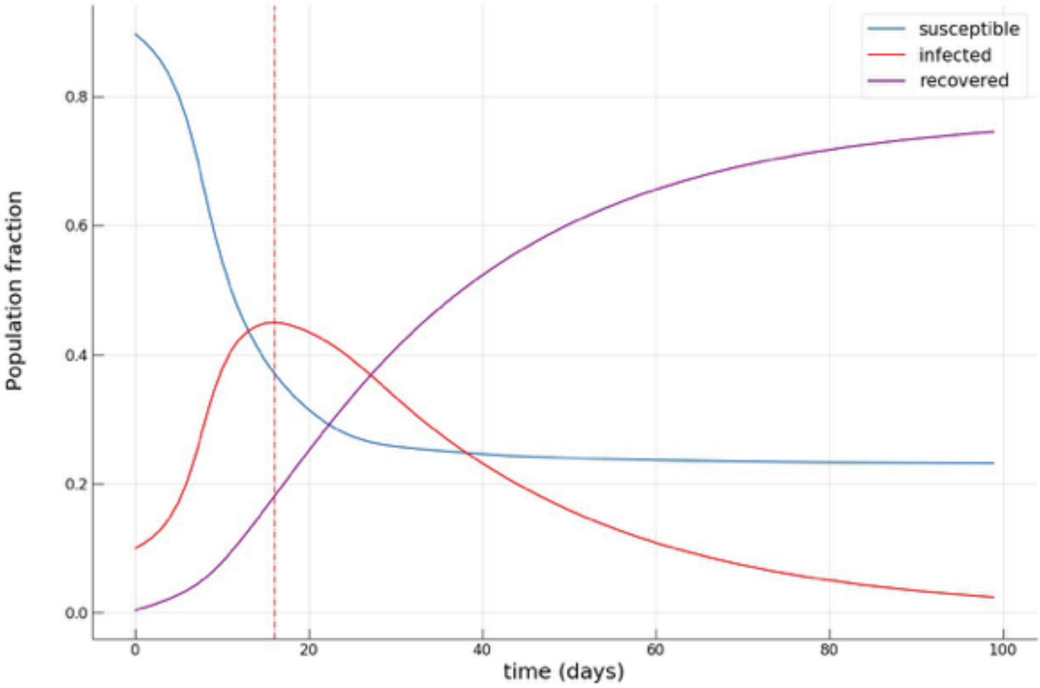
Reduce public transport?

For our first simulation, we will imagine a sustainable public transport-dominated future urban mobility with *α=0.9*:



We see how fast the infected fraction of the population is climbing up immediately, reaching the epidemic's peak on around day 8–10, with **almost 70% of the population**

infected, while only a small portion (~10%) of the population having recovered from the disease. Towards day 100, when the epidemic has receded, we see **the fraction of recovered individuals reach a staggering 90%**! Now let's see if reducing the intensity of public transport travel to something like $\alpha = 0.2$ has any effect on mitigating the epidemic spread. This can either be interpreted as **taking drastic measures to reduce urban mobility (e.g., by issuing a curfew)** or as **increasing the share of private car travel to reduce chances of infection *during* the travel**.

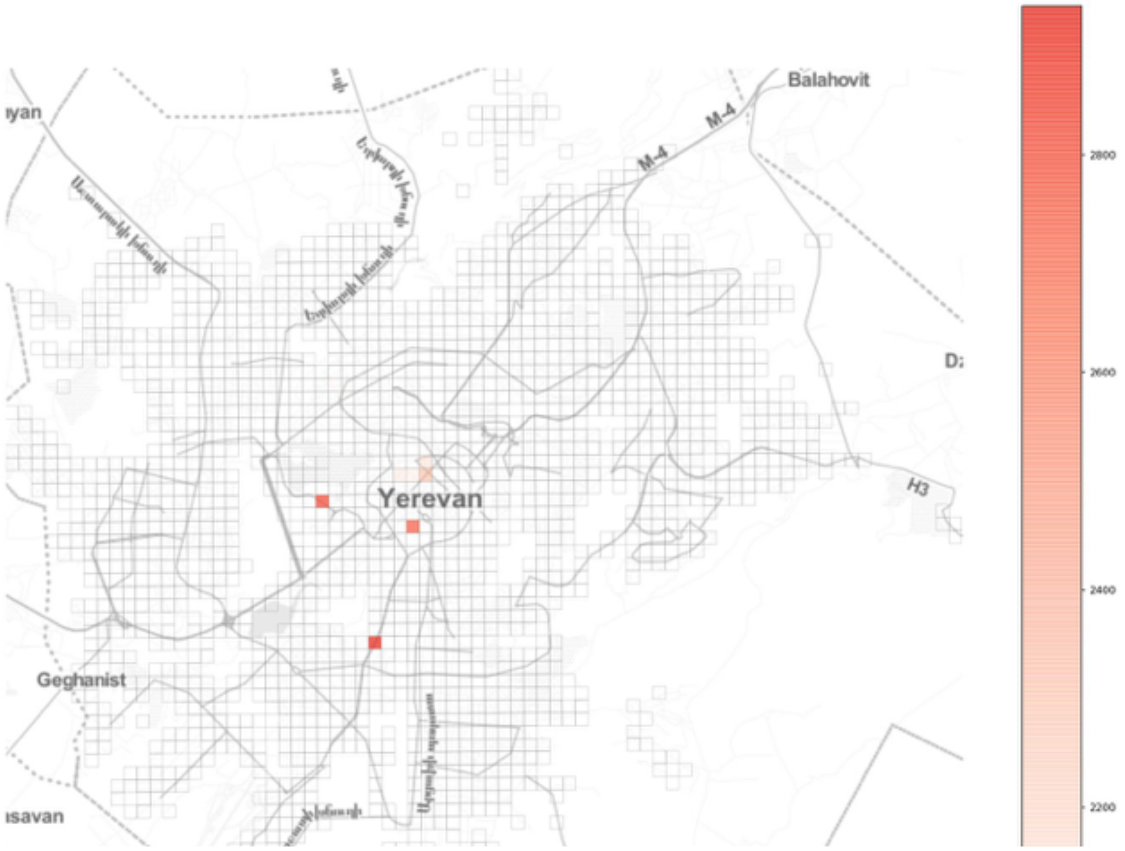


We see how the peak of the epidemic comes somewhere between day 16 and 20, with a **significantly smaller infected**

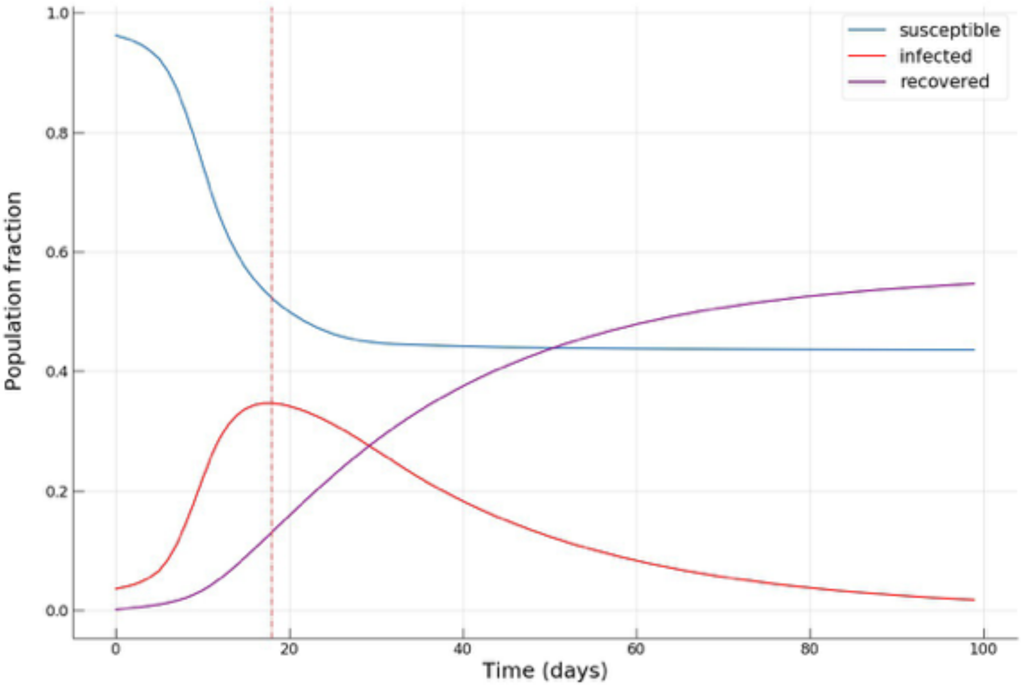
group (~45%) and twice as many recovered (~20%). Towards the end of the epidemic, the fraction of susceptible individuals is also twice as big (~24% vs. ~12%), meaning that more people have escaped the disease. As expected, **we see that the introduction of dramatic measures to temporarily bring urban mobility down has a big impact on the disease spreading dynamics**.

Quarantine popular locations?

Now, let's see whether another intuitive idea of completely cutting off a few key popular locations has the desired effect. To do this, let's pick the locations associated with the upper 1 percentile of mobility flows,

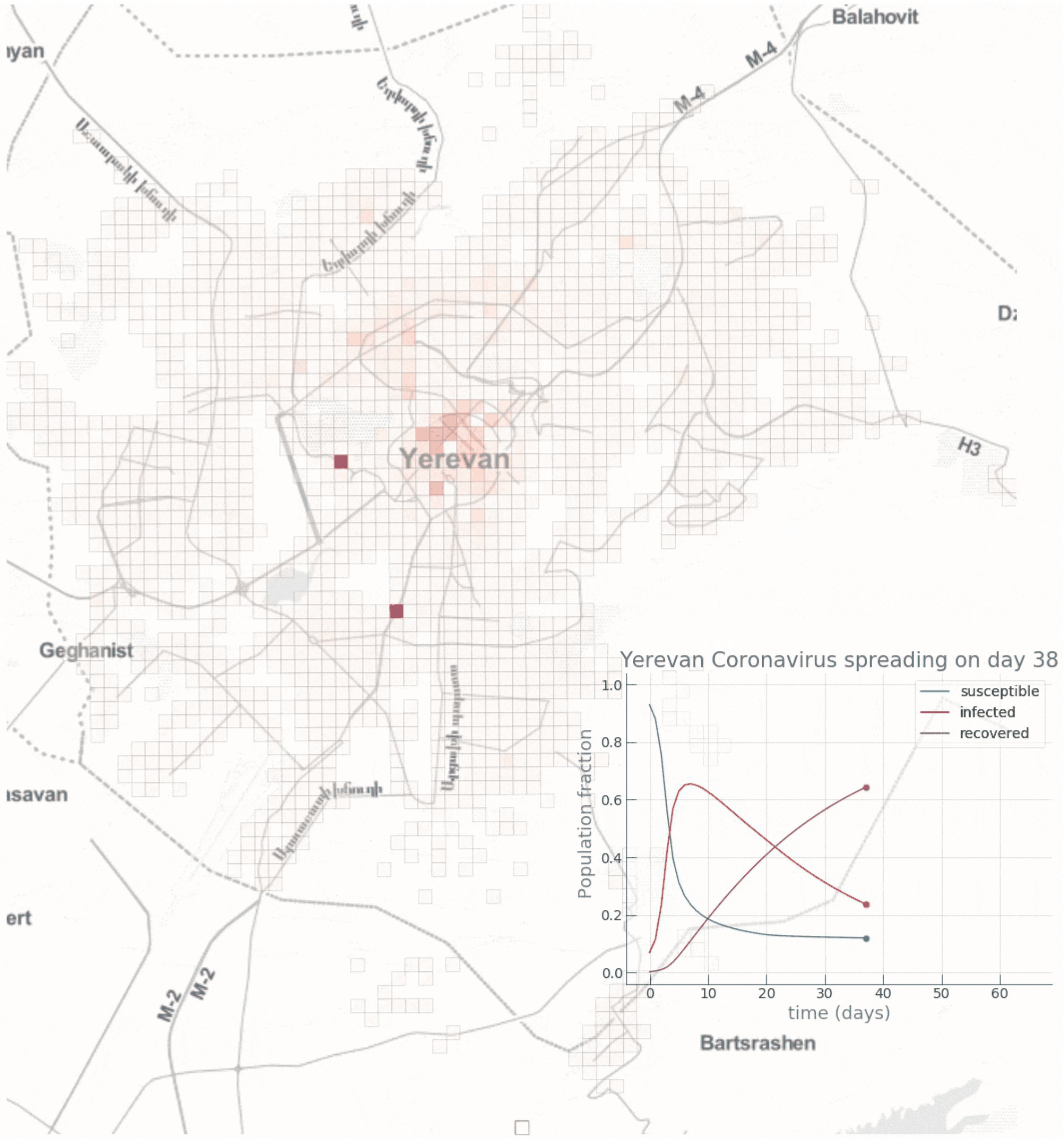


and **completely block all flow to and from those locations**, effectively establishing there a quarantine regime. As we can see from the plot, in Yerevan these locations are mostly in the city center, with two other locations being the two largest shopping malls. Choosing a moderate $\alpha = 0.5$, we obtain:



We see an even smaller fraction of infected individuals at the epidemic's peak (~35%), and, most importantly, we see that towards the end of the epidemic, **around half of the population remains susceptible, effectively escaping from contracting the infection!**

Here is a small animation visualising the dynamics of the high public transport share scenario:



Conclusion

By no means claiming accurate epidemic modelling (or even any substantial knowledge in epidemiology beyond the basics), our aim in this post was to get a first insight on how network effects come into play in an urban setting during an infectious disease outbreak. With ever-increasing population densities, mobility, and dynamics, our cities become more exposed to "black swans" and become more fragile. And since **you can't fetch the coffee if you're dead, smart and sustainable cities will be meaningless without effective and efficient crisis handling capability and mechanisms.** For instance, we saw that the introduction of quarantine regimes in key locations, or taking draconian measures to curb mobility, can be instrumental during such a health crisis. However, a further important question would be **how to implement such measures while minimizing damage and loss to the functioning of the city and its economy?**

Further yet, the exact epidemic spreading mechanisms of infectious diseases are still an active area of research and the advances in this fields will have to be communicated to and integrated in urban planning, policymaking, and management to make our cities safe and antifragile.

P.S. Read the original post here.

The code for the above simulations:

```
import numpy as np
  # initialize the population vector from the origin-
destination flow matrix
  N_k = np.abs(np.diagonal(OD) + OD.sum(axis=0) -
OD.sum(axis=1))
  locs_len = len(N_k)                 # number of
locations
  SIR = np.zeros(shape=(locs_len, 3)) # make a numpy
array with 3 columns for keeping track of the S, I, R
groups
  SIR[:,0] = N_k                      # initialize the S
group with the respective populations

  first_infections = np.where(SIR[:, 0]<=thresh, SIR[:,
0]//20, 0)   # for demo purposes, randomly introduce
infections
  SIR[:, 0] = SIR[:, 0] - first_infections
  SIR[:, 1] = SIR[:, 1] + first_infections
# move infections to the I group

  # row normalize the SIR matrix for keeping track of
group proportions
  row_sums = SIR.sum(axis=1)
  SIR_n = SIR / row_sums[:, np.newaxis]

  # initialize parameters
  beta = 1.6
  gamma = 0.04
  public_trans = 0.5                                 #
alpha
  R0 = beta/gamma
  beta_vec = np.random.gamma(1.6, 2, locs_len)
  gamma_vec = np.full(locs_len, gamma)
  public_trans_vec = np.full(locs_len, public_trans)

  # make copy of the SIR matrices
  SIR_sim = SIR.copy()
  SIR_nsim = SIR_n.copy()

  # run model
  print(SIR_sim.sum(axis=0).sum() == N_k.sum())
  from tqdm import tqdm_notebook
  infected_pop_norm = []
  susceptible_pop_norm = []
  recovered_pop_norm = []
  for time_step in tqdm_notebook(range(100)):
      infected_mat =
np.array([SIR_nsim[:,1],]*locs_len).transpose()
      OD_infected = np.round(OD*infected_mat)
      inflow_infected = OD_infected.sum(axis=0)
      inflow_infected =
np.round(inflow_infected*public_trans_vec)
      print('total infected inflow: ',
inflow_infected.sum())
      new_infect = beta_vec*SIR_sim[:,
0]*inflow_infected/(N_k + OD.sum(axis=0))
      new_recovered = gamma_vec*SIR_sim[:, 1]
      new_infect = np.where(new_infect>SIR_sim[:, 0],
SIR_sim[:, 0], new_infect)
      SIR_sim[:, 0] = SIR_sim[:, 0] - new_infect
      SIR_sim[:, 1] = SIR_sim[:, 1] + new_infect -
new_recovered
      SIR_sim[:, 2] = SIR_sim[:, 2] + new_recovered
      SIR_sim = np.where(SIR_sim<0,0,SIR_sim)
      # recompute the normalized SIR matrix
      row_sums = SIR_sim.sum(axis=1)
      SIR_nsim = SIR_sim / row_sums[:, np.newaxis]
      S = SIR_sim[:,0].sum()/N_k.sum()
      I = SIR_sim[:,1].sum()/N_k.sum()
      R = SIR_sim[:,2].sum()/N_k.sum()
      print(S, I, R, (S+I+R)*N_k.sum(), N_k.sum())
      print('\n')
      infected_pop_norm.append(I)
      susceptible_pop_norm.append(S)
      recovered_pop_norm.append(R)
```

Data Science Python Modeling Urban Planning Epidemic

Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just $5/month. Upgrade

About Help Legal