LINMA2370 - Appendix to the project (part 1)

Benjamin Chiêm
`benjamin.chiem@uclouvain.be`

October 7, 2020

Introduction

In order to simulate the spread of an epidemic on a network of contacts between individuals, we need specific algorithms. Instead of dealing with deterministic differential equations, we are going to run *stochastic* simulations where discrete events (e.g. getting infected or recovering from the disease) happen randomly with some probability and in continuous time. Since these algorithms are not part of the course material, we will use publicly-available toolboxes implementing them and allowing you to flexibly investigate different spreading scenarios.

1 Installing the toolboxes

We will use two toolboxes :

1. **NetworkX**, to generate and manipulate networks
2. **Epidemics on Networks (EoN)**, to run the simulations

The easiest way to install these toolboxes is through the Python package installer `pip`. This tool is most probably already installed on your computer ; otherwise, follow the instructions described here : `https://pip.pypa.io/en/stable/installing/`

To check whether pip is correctly installed, try to run the following command (the part in parentheses is for Windows users only)

- (`python -m`) `pip -V`

in the command line[1] to display the version of pip.

Once pip is installed, run the following commands in order to respectively install NetworkX and EoN (pay attention to uppercases and lowercases):

- (`python -m`) `pip install networkx`
- (`python -m`) `pip install EoN`

To check your installation, run the code that is provided with the statement of the project and observe the results.

[1]Either the command prompt ("Invite de commandes" in French) or the Anaconda prompt on Windows, depending on your Python installation. You can use the Terminal on MacOS and Linux systems.

2 Networks and stochastic simulations

A network (or a graph) is a collection of nodes (or vertices) connected by edges. In our context, nodes represent individuals and an edge is present between two nodes if the corresponding individuals are in contact ; see Figure 1.

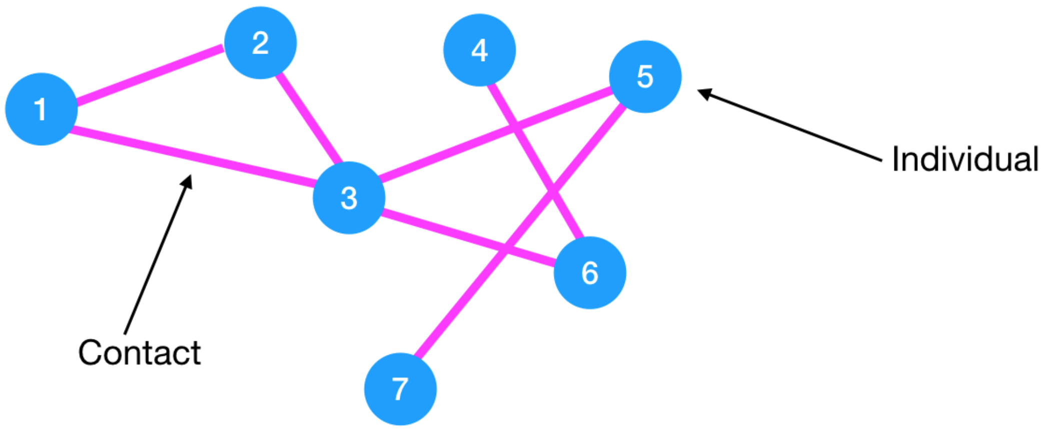

Figure 1: A contact network. Nodes (individuals) are shown in blue, edges (contacts) are shown in purple.

For simplicity, we will only consider undirected and binary graphs, i.e. contacts are reciprocal and have no intensity scale. A possible way to mathematically represent a graph is via its adjacency matrix $\mathbf{A}$. In a graph of N nodes, the elements $\mathbf{A}_{ij}$ of the $N \times N$ matrix $\mathbf{A}$ are defined as follows:

- $\mathbf{A}_{ij} = 1$ if nodes i and j are connected
- $\mathbf{A}_{ij} = 0$ otherwise

For instance, the adjacency matrix of the graph shown in Figure 1 is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The degree of a node is its number of neighbors. We can derive the degree d_i of node i from the adjacency matrix :

$$d_i = \sum_{j=1}^{N} \mathbf{A}_{ij}$$

The degree distribution of a graph can be represented by the histogram of the degree of all nodes in the graph. In the NetworkX toolbox, a graph `G` is a Python object with several attributes and methods. For example, `A = networkx.adjacency_matrix(G)` returns the adjacency matrix[2]

[2]Beware that this will return a SciPy sparse matrix. You may want to manipulate the full matrix using `A_full = A.todense()`

`A` of the graph `G`.

When simulating an epidemic on a network, each node can either be susceptible, infected or recovered. In order to simulate the stochastic (Markovian) spread, a popular approach is the Gillespie algorithm. In short, the algorithm can be decomposed into 4 steps:

1. Compute the probability of each possible event (e.g. node i becomes infected, node j recovers, etc) with respect to the transmission rate, the recovery rate and the topology of the network
2. Compute the waiting time until the next event
3. Select which event occur, with respect to the probabilities previously computed
4. Update the status of the nodes and repeat

In the code provided with the statement, we used a slightly different implementation (*event-driven* algorithm) that is faster in general. However, both algorithms simulate the same process and will provide comparable results (more information in the textbook ; see below). Note that in the EoN toolbox, the initial condition can be determined either by a fixed proportion of randomly chosen nodes that are initially infected, or by a list of initially infected nodes provided by the user.

3 Using the documentation

The practical documentation for both toolboxes is available on their respective webpage :

- `https://networkx.github.io/documentation/stable/`
- `https://epidemicsonnetworks.readthedocs.io/en/latest/index.html`

There, you will find details about the functions that you can use, their signature, example codes as well as possible troubleshooting tips.

The EoN toolbox comes with the following textbook (not part of the course material):

- István Z. KISS, Joel C. MILLER & Péter L. SIMON (2017). Mathematics Of Epidemics On Networks: From Exact to Approximate Models. Springer.

For those who are interested, the online version of this textbook is available via the UCLouvain network : `https://link.springer.com/book/10.1007%2F978-3-319-50806-1`

Remarks

If you have troubles installing the toolboxes or running the provided code, please contact Benjamin Chiêm (benjamin.chiem@uclouvain.be) as soon as possible. You can also ask for an appointment if you need help with the scenario that you want to implement (e.g. generating specific network structures, using different initial conditions, etc). Due to the sanitary restrictions in the context of the Covid-19 pandemic, appointments will by default take place on Microsoft Teams, up to some exceptions for obvious reasons (e.g. installation troubleshooting).