Coronavirus-Basic-Model

A basic mathematical tool for predicting the spread of the Covid-19 pandemic, using the SIR model for infectious diseases. It was created by Chistos Frantzolas and Ilias Xenogiannis. This project can run with versions of Python above 3.5.2. You can install the complete requirements by using the command 'pip install -r requirements.txt'. [Extra 1]

Disclaimer: Neither I nor anyone that currently works on this project is an epidimiologist. This project has been built for purely academic purposes.

SIR Model

Compartmental models are a technique used to simplify the mathematical modelling of infectious disease. The population is divided into compartments, with the assumption that every individual in the same compartment has the same characteristics. [1]

The SIR model divides the population into 3 groups: S = S(t) is the number of susceptible individuals, I = I(t) is the number of infected/infectious individuals, and R = R(t) is the number of recovered/removed individuals. [2]

We make the assumptions that: 1. No one is added to the susceptible group, since we are ignoring births and immigration. 2. The time-rate of change of S(t), the number of susceptibles depends on the number already susceptible, the number of individuals already infected, and the amount of contact between susceptibles and infecteds. In particular, suppose that each infected individual has a fixed number b of contacts per day that are sufficient to spread the disease. 3. We also assume that a fixed fraction k of the infected group will recover during any given day. [2]

The differential equations that describe the model are as follows: [1]

You can find more information about the SIR model in the resources [3], [4].

Note that there are 2 parameters we don't know the values of (β, γ, presented as b and k in the code respectively). We can estimate the number of days each person remain infectious and thus have the recovery rate be the inverse of this period. On the other hand, there are studies on the R0 (basic reproductive rate) of the novel coronavirus, which is the number of people an infectious person is going to transmit the virus to in total. The R0 of Covid-19 is estimated to be 2.06 to 2.52. [5] Then, we can calculate b, with the equation:

Of course, b is not going to remain constant. Preventative measures like social distancing, lockdowns and better hygiene limit possible contacts that end up transmitting the virus, thus reducing b. On the other hand, environmental factors can also have an impact on the transmission rate of an infectious disease, although it's not yet certain how this will affect the Covid-19 epidemic.

Scrapping Data

The population data and the Covid-19 confirmed cases and deaths data is taken from Worldometers.

More specifically, the population data by country, along with data for age, urbanization, density etc. are taken from [6]. The total confirmed cases of Coronavirus for the entire planet is taken from the "Total Cases" graph on [7], the daily total death toll from the "Total Deaths" graph on [8] and the active cases day-by-day are taken from the "Active Cases" graph on [9]. Finally, data for each specific country is taken from their respective graphs in each country's page, if there is one. p.e. China [10]

The graphs on the site is updated approximately every 24 hours. Thus, if the model runs a scenario it hasn't run in the last 30 hours it will update its data base (simple text files for the time being). In any other case, it will simply read data from the corresponding files.

In order to understand the scrapping function you need to know some basic things about the regular expressions module of Python 3 [Extra 2]

Running the Model

You can run the model from the main file "sir_prediction.py". Initialize your scenario and data with the `init_data()` function, run the model with `sir_method()` and plot them with the function `plotting()`. There is extensive documentation on the py files themselves, but we can walk through an example here.

Initializing Data

The init_data function takes a single keyword argument `scenario` with a default value "world". There are scenarios for many countries, like "US" for the United States, "China" or "Greece". The way that countries are named in the url of the worldometers website and the demographic data table differs in some cases. If a country's data do not load properly you most likely have to create a special case for it. There is also the scenario "without China" which excludes Chinese cases from worldwide data, as the country is an "outlier", both by being hit first, and also taking extraordinary measures pretty early. The data variable that the `init_data()` function returns is a Python Dictionary. You can find its keys and structure in the documentation.

Running the model

The sir_method function, takes three arguments; data, the same python dictionary that the init_data() function outputs, offset and run. The offset argument specifies how many days before the present date should the model start its prediction (an offset of one means that the model starts running from the current date. The argument run specifies for how many days after the offset date should the model make predictions for. The output is the updated Data dictionary.

Plotting the results

The `plotting()` function takes as arguments the data to be plotted, the list "selection" that should contain the dictionary keys of the data that you want to plot, and the "scale" keyword argument, which specifies the scale of the y axis (either "linear" or "log")

Example

Let's see an example of the "without China" scenario, which begins prediction 25 days from the present, to the present day:

```
 Data = init_data(scenario="without China")
Data = sir_method(Data, offset=25, run=25)
plotting(Data, ["Total list", "Deaths list", "I list"], scale='linear')
```

The output plot is this, on the 30th of March 2020:

Image of results plot

Ideas for the Future

As the project is in its very early stages, there is much room for improvement. You can contribute to it by: 1. Improving the data collection or expanding it through other sources to encompass things like: 1. Measures taken by governments. 1. Hospital capacity in each country. 1. Deviding populations by groups. 1. Testing numbers. 1. Dividing countries into smaller areas. 1. Improving the model itself: 1. Figure out different trends in the data. 1. Calculate the transmission rate in different conditions (climate, measures etc.) 1. Use different models to simulate the effects of different preventative measures. 1. Improving the presentation of the simulation. 1. Create an animation through time. 1. Make maps that demonstrate the spread of the disease. 1. Create a (G)UI for the users to initialize the model without interracting with the code itself. 1. And anything else that comes to mind!

FAQ

Sources

1. https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology
2. https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model
3. https://www.youtube.com/watch?v=k6nLfCbAzgo
4. https://www.youtube.com/watch?v=gxAaO2rsdIs
5. https://www.ijidonline.com/article/S1201-9712(20)30091-6/fulltext
6. https://www.worldometers.info/world-population/population-by-country/
7. https://www.worldometers.info/coronavirus/
8. https://www.worldometers.info/coronavirus/coronavirus-death-toll/
9. https://www.worldometers.info/coronavirus/coronavirus-cases/
10. https://www.worldometers.info/coronavirus/country/china/

Extra

1. https://pip.pypa.io/en/latest/user_guide/#requirements-files
2. https://docs.python.org/3/library/re.html