Covid-19, data and a simple compartment model

Summer 2021

Aug 19, 2021

Learning objectives. By completing this project, the student will:

- Learn to import data into pandas
- Write simple functions to generate plots, and inspect data visually
- Constrain model parameters by comparing with data
- Use a model to interpret data, and reflect on strength and weakness of the approach

Project overview. In this project we are modeling the first 250 days of the Corona outbreak. The model we are using are the simplest possible model, and you are encouraged to reflect on the limitations. All modeling projects consists (usually) of two parts, the first part is to get an overview of the data, and the second part is to develop a suitable model that can be compared or tuned to parts of the data.

1 Covid-19 Data

The data used in this project are date published at Johns Hopkins corona virus resource center[1] - a screen shot can be seen in figure 1.

Data for the Corona virus are readily available. We will use data found at the Github repository Center for Systems Science and Engineering (CSSE) at Johns Hopkins University[2]. We have already extracted country-level data for you, and stored it in a processed format in the text file `data/corona_data.dat`. Data for the Hubei province in China, where it is believed that the virus first arose, is included in the text file.

[1] `https://coronavirus.jhu.edu/map.html`
[2] `https://github.com/CSSEGISandData/COVID-19`

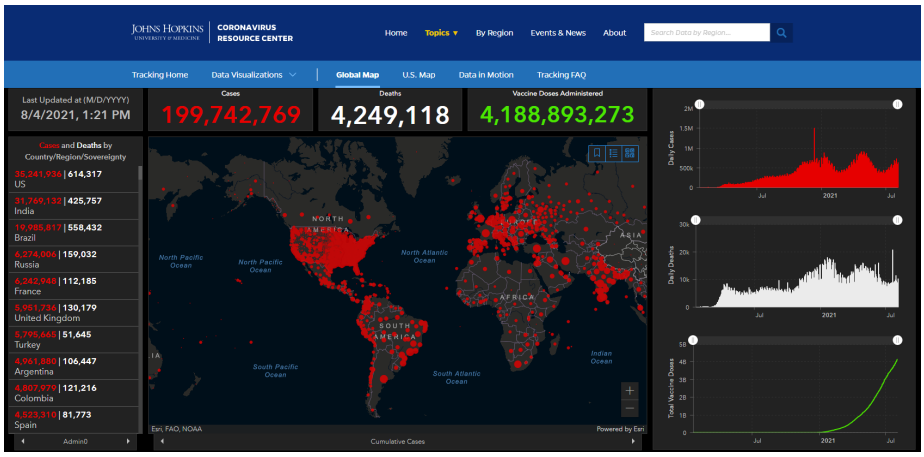

Figure 1: A screenshot of the Johns Hopkins corona virus resource center.

The reason for doing so is that while the original data were organized by date since January 22 2020, we would like to plot the data versus the time of *the first confirmed case*. This makes it easier to apply the same model to different locations.

2 Exercise 1: Loading data into data frames

Next, we want you to get familiar with the data, the relevant file is located in a separate folder `data`. We are going to use pandas[3] to load the data. Pandas is a very flexible library, but we will only use it in a very limited way.

Copy and paste code.

Copy and paste code from various sources is usually a good way to get started, i.e. just to solve the task at hand. We all do it. However, if you want to grow and develop as a programmer, you need to understand all the steps in the code (or code snippets) you have copied, *and not only the end result*. Understanding all the steps in the code will make you able to identify unnecessary steps, and therefore simplify your code and make it "mean and lean".

Use of libraries.

There are many Python libraries out there, but a good advice is to use no more libraries than necessary. For scientific computing there are a few libraries that are used by almost everyone, the most important being NumPy[a]. `Numpy` is based on well-optimized C code and is designed to perform heavy numerical computations faster than native Python. If you see code that uses a different library to achieve something, ask yourself if the same thing could be done with `Numpy`. Another important reason to prefer `Numpy` is that the syntax is known to almost everyone, which makes it quite easy for other people to read and understand your code by visual inspection.

Of course, some tasks are best performed by other libraries. As you will see in this introductory project, pandas[b] lets you read data from files using very few lines of code. When doing more advanced tasks with the file system on your computer, the pathlib library[c] is very useful, as is the os module[d]. For plotting, matplotlib[e] is where most Python programmers start.

[a] `https://numpy.org`
[b] `https://pandas.pydata.org/`
[c] `https://docs.python.org/3/library/pathlib.html`
[d] `https://docs.python.org/3/library/os.html`
[e] `https://matplotlib.org`

Part 1. Execute the following line of code

[3] `https://pandas.pydata.org/`

```
df = pd.read_csv('../data/corona_data.dat',sep='\t')
```

- This command reads the content of `corona_data.dat` into a DataFrame[4], a DataFrame is basically a two dimensional labeled data structure very similar to an excel sheet. For more information you can check out the documentation[5]
- What happens if you remove `,sep='\t'`, why?
- Compare `print(df)` with what you see if you open the file `corona_data.dat` in a text editor.

Part 2. The `df` DataFrame contains all COVID-19 data from every country, run the following code

```
df=df[df['LOCATION'] == 'Afghanistan']
```

- explain what the code does.

A good starting point is to start with the innermost statement and work from there, i.e. `df['LOCATION'] == 'Afghanistan'`

Part 3. Write a Python function that takes one argument which is the name of the province or country, and one *default* argument which is the name of the data file where the COVID-19 data are located. The function should return the DataFrame containing the data from that country.

3 Exercise 2: Visualizing data

Although pandas have built in plotting, we suggests that you stick to Matplotlib[6]. To visualize data, it is best to have the $x-$, and $y-$ data stored in Numpy arrays.

Part 1.

- Use the Pandas function `to_numpy`[7] to extract the columns `ELAPSED_TIME_SINCE_OUTBREAK` and `CONFIRMED` in two separate Numpy arrays for a given country, e.g. Norway.

[4] `https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html`
[5] `https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html`
[6] `https://matplotlib.org/`
[7] `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_numpy.html`

Part 2.

- Assuming that the data is stored in two Numpy arrays, e.g. `time` and `confirmed`, make a 2d scatter plot[8] of the data.

Part 3. Write a function that takes one argument which is the name of a province or country, and one *default* argument which is the name of the data file holding the COVID-19 data. The function should produce a scatter plot for the actual province or country. A good tip when writing this second function is to re-use the one you made previously!

Avoiding code duplication.

It is good programming practice to avoid copying & pasting the same (or very similar) code many times over in your overall program. A good way to reduce code repetition is to define smart functions that you can call in different places. Whenever you make a change to the function, all places in the code making use of that function is automatically updated!

However, it can be challenging to design good functions. A rule-of-thumb is that any one function should do one thing very well, but not have too many responsibilities. For instance, when working with data it is almost always a good idea to separate the reading and pre-processing of data from code that uses the data for modelling, plotting etc.

Part 4.

- Extend the function you wrote above to include a suitable title and axis labels (this is especially important for understanding what you are plotting!)

Note that it is possible to return figures from functions, which makes it possible to further customize your plots even after calling the plotting function!

Part 5.

- Make a plot showing the number of confirmed cases since the time of the first outbreak in a) Hubei and b) Norway.

Figure 2 shows a possible solution.

[8] `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html`

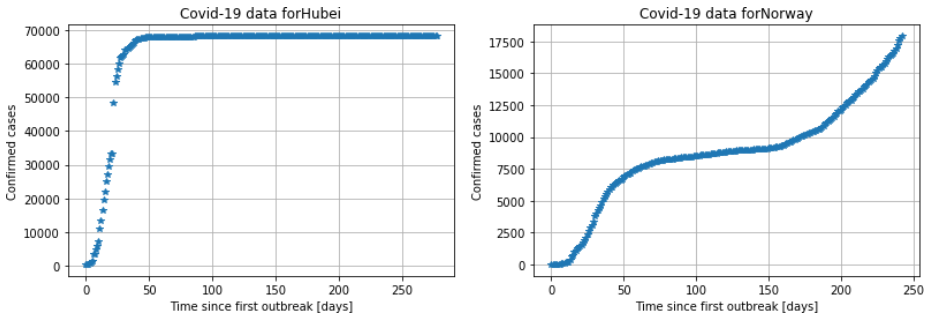

Figure 2: Covid-19 data for Hubei and Norway.

4 Modeling Covid-19: The SI-model

Compartment models [2] are widely used to study how an epidemic disease might spread in a population. In these models, the total population is partitioned into compartments based on a set of possible "disease states", and differential equations are set up to describe how individuals "flow" from one compartment to another. The equations can be either deterministic or stochastic. While the latter type of model is more realistic, we will only study deterministic models in this project.

We first consider the SI-model, which consists of only two compartments:

1. S - Susceptible: people at risk of infection.
2. I - Infected.

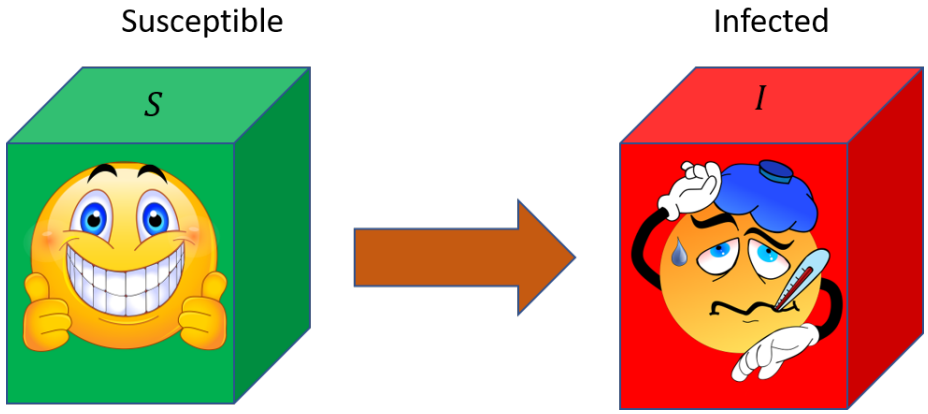

Figure 3: The SI-model. Note that only transport from the class of exposed humans to sick are allowed.

Let N denote the total population size. For each time t, let $S(t)$ denote the number of susceptible people, and $I(t)$ the number of infected people. To develop a model, we need to calculate the rate of flow between the two compartments in figure 3. We start by making some observations:

- During each time interval Δt, a certain number of individuals will come into contact with each other.
- We only care about healthy (susceptible) - sick encounters, i.e. a sick person cannot infect another sick person.
- Whenever a healthy person meets a sick, there is a certain probability that the person becomes infected.

We shall take our imagined population to be *well mixed*, meaning that pairs of individuals interact with equal probability. Let $\mathcal{C}(N)$ denote the rate at which *any* individual in the population contacts *any* another individual, i.e., the average number of contacts made per unit time. Then, we can estimate the change in the healthy population from time t to $t + \Delta t$ as:

$$S(t + \Delta t) - S(t) = -\mathcal{C}(N) \cdot \Delta t \cdot p \cdot q \cdot S(t)\,. \tag{1}$$

where p denotes the conditional probability that a given contact is between a susceptible and infected individual, and q is the probability that such an encounter leads to disease transmission. Because of the well-mixed condition, a good assumption is that $p = I(t)/N$; thus, the challenge consists in estimating $\mathcal{C}(N)$ and q. In principle, both of these parameters may vary in time, but for now we shall regard them as constant. By merging them into a single factor, β, we get

$$S(t + \Delta t) - S(t) = -\beta \cdot \Delta t \cdot \frac{S(t)I(t)}{N}\,, \tag{2}$$

Finally, by dividing by Δt and letting $\Delta t \to 0$, we obtain the following ODE:

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = -\beta \cdot \frac{S(t)I(t)}{N}\,. \tag{3}$$

Similarly, the evolution of the sick population is given by:

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = +\beta \cdot \frac{S(t)I(t)}{N}\,. \tag{4}$$

How to interpret β?.

By saying that β is constant, we have made two very strong assumptions:

- People make the same number of contacts regardless of the population size, and independent of time.

- The probability of becoming sick, given that you meet an infected person, is always the same.

In reality, β is time-dependent, as it implicitly accounts for a lot of biomedical, physical, and sociological factors. For example, in the beginning of an outbreak, β is likely to be large, because people might not yet understand the severity of the situation, or they may be in denial. As people start to realize the danger and fight back against the disease, β will most likely decrease.

5 Exercise 3: [OPTIONAL] Analytical solution

Clearly, if there are no infected individuals at time zero, the above equations predict that nothing will happen later either. We shall therefore assume that the initial number of sick is close to one; typically $I_0 = I(0) = 1$.

Part 1.

- Show that the analytical solution to the SI-model, equations (3) and (4), is

$$S(t) = \frac{(S_0 + I_0)\frac{S_0}{I_0}\exp(-\beta t)}{1 + \frac{S_0}{I_0}\exp(-\beta t)}\,, \tag{5}$$

$$I(t) = \frac{S_0 + I_0}{1 + \frac{S_0}{I_0}\exp(-\beta t)}\,, \tag{6}$$

where $S_0 = S(0)$, and thus $S_0 + I_0 = N$. Hint: replace $I(t)$ with $N - S(t)$ in equation (3).

6 Exercise 4: Is the model any good?

"All models are wrong, but some are useful" is a famous quote by G. E. Box [1]. The only way we can investigate if a model is useful is to compare it with data. Clearly, the SI-model presented above is very simple - it contains only a *single* model dependent parameter β. But, can it be useful? i.e. can we learn something about the spread of the Corona virus and can this insight be used to take e.g. preventive measures?

Part 1.

- Make a Python function that solves equations (5) and (6).
- The function should take the following input arguments: a Numpy array of times, the initial number of susceptible people, the initial number of infected people, and a value for β.

- The function should return two Numpy arrays, where the first array holds the vector of susceptible individuals at each time step, and the second array holds the vector of infected people.

The following lines of code exemplifies how such a function could be used by calling code:

```
S,I = calc_SI_model(report_times, S0, I0, beta)
```

Part 2.

- Write a function that plots the analytical solution to the SI-model together with the actual number of confirmed cases observed for a given country.
- Try to re-use several of the functions you made previously when implementing this function.

Part 3. Let us first consider the Hubei province in China where the city of Wuhan is located.

- Assume for simplicity that $I_0 = 1$, and adjust S_0 and β to investigate if it is possible to match the observed behavior of the spread of the corona virus with the SI model.

Part 4. In the database there are also data from the cruise ship with corona outbreak "Diamond Princess".

- Put $S_0 = 700$ to match the number of people on the boat, and use the same β value you found for Hubei. What do you observe? (You should be able to generate curves similar to the ones shown in figure 4).

Part 5.

- Do the same for Norway. What value of β do you need to use in order to find a reasonable match for Norway?
- Does the model match the data better or worse in this case?

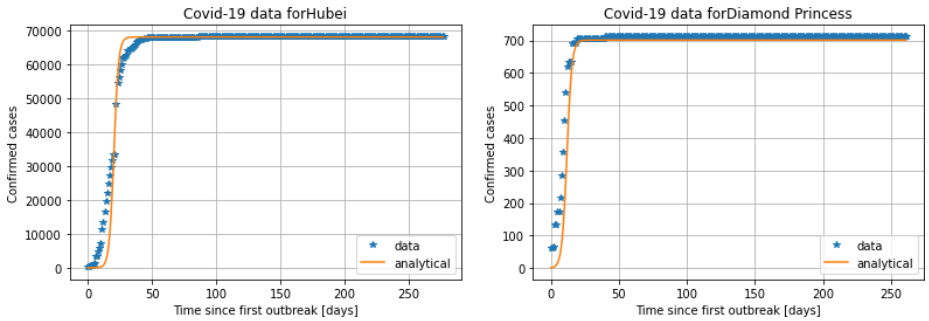

Figure 4: Covid-19 data and model for Hubei and Diamond Princess.

7 Exercise 5: Extending the model

From the last exercise you might have noticed that the SI model works quite good when the disease spreads fast in a smaller area and in short time. However a clear weakness with the model is that we tune the number S_0 to the data. In reality S_0 should be the total population of a country, or at least the total number of people in a city where there is a large outbreak. In Wuhan there are 11 million people, and in Hubei there are 58.5 million people. Setting S_0 to 11 or 58.5 million would greatly exaggerate the number of infected people in our model. The problem is clearly the probability of infection β. As measures are taken β should decrease as a function of time. To capture this behavior, the covid-infection rate will now be assumed to decline exponentially:

$$\beta(t) = \beta_0 e^{-\lambda t}. \tag{7}$$

We can still use the analytical solution in equation (5) and (6), but we have to replace

$$\beta t \to \int_0^t \beta_0 e^{-\lambda t} dt = \frac{\beta_0}{\lambda}(1 - e^{-\lambda t}). \tag{8}$$

In the above expression, β_0 is the initial infection rate, whereas λ plays the role of measures taken. A high value of λ is indicates strong disease-prevention measures.

Part 1.

- Make a function that plots the number of confirmed cases for a given country, as well as the extended analytical solution with β_0 and λ as input parameters.
- Instead of creating a completely new function, you might want to adapt your previous one (e.g., by using $\lambda = 0$ as default argument).

Population data can be found in the `data` folder.

Part 2.

- Tune λ and β_0 to match the Hubei data. This is quite difficult, but a strategy is to first adjust β_0 to match the initial rise in the confirmed cases (For Hubei, a value for β_0 close to 2 gives a decent match). Next, adjust λ to match the data.

Part 3.

- Use the $\beta-$ and $\lambda-$ values you found for Hubei to predict the disease outbreak in Norway (You need to reduce S_0 to 5.4 million, which is the population of Norway). What do you observe? Do the results match your expectations?

Part 4.

- How do you need to change λ and β to improve the match for Norway? If we assume that the value of λ represents the government response, does the different values (λ_{Norway}, λ_{Hubei}) match your expectations?

References

[1] George EP Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

[2] William Ogilvy Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.