Covid-19, data and a simple compartment model

Aksel Hiorth and Oddbjørn Nødland

Aug 19, 2021

How to write an abstract.

The abstract should focus on the *key* ideas in the report, *key* results, and implications of your findings.

Abstract

In this project, a simple SI (Susceptible, Infected)-model is used to analyze the first 250 days of the Corona virus outbreak. In areas with high population density, a model with a constant disease transmission rate, β, fits the data well, provided we match the total number of infected people to the data. In areas with a lower population density, as in the Nordic countries, the same model works poorly. Extending the model to allow for a time-dependent β makes it possible to account for varying governmental responses among countries. The results presented here supports the notion that the governmental response was stronger in Hubei, China compared to Norway.

How to write the introduction.

Use the introduction to place your work in context, and outline your approach to the problems under investigation. It is good to cite some appropriate references here, but make sure that you do not copy directly from other sources; that will be regarded as cheating! You will lose points and get a lower grade.

Also, the introduction should *not* be a copy of the handed-out exercise text, instead try to use your own words as much as possible.

1 Introduction

The Corona virus disease, COVID-19, is brought on by infectious contact with *SARS-CoV-2, severe acute respiratory syndrome coronavirus 2* [6]. Since its original discovery in Wuhan, China [9], it spread rapidly to most of the world’s countries and has cause a pandemic we are still in the middle of.

In this project we investigate the 250 first days of the Corona virus outbreak. The project is divided into two parts. In the first part, we import and view data on reported number of confirmed cases using the Python libraries `pandas` and `matplotlib`. The data can be freely downloaded from the Johns Hopkins git repository, Center for Systems Science and Engineering (CSSE) at Johns Hopkins University[1]. In the second part, we use a mathematical compartment model to interpret the data. Finally, we present our conclusions as well as reflections about what we learned working on the project.

2 Exercise 1

Part 1. We start by reading the contents of the text file `corona_data.dat` into a `pandas.DataFrame` object:

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

In the text file, columns of data are separated by a tab (`\t`) marker. Therefore, we have to use the command `sep=’\t’` in order for the `read_csv` function to work properly (as explained in the documentation[2])

Part 2. The data frame, `df`, now contains COVID-19 data from all countries, as well as several regions and other kinds of locations. If we want to extract only the data for a specific country, say Afghanistan, we can also do that very easily with `pandas`:

```
df = df[df[’LOCATION’] == ’Afghanistan’]
```

The innermost command (`df[’LOCATION’] == ’Afghanistan’`) returns an array of boolean variables (i.e., either `True` or `False`) signifying which entries have "Afghanistan" in the location column. Since `pandas` supports boolean masking[3], this array can in turn be passed to the original data frame, giving back a sub-dataframe containing only the rows corresponding to `True`.

Part 3. Since we are interested in comparing data for different locations, we create a Python function that returns the data for one specific location:

```
def get_corona_data(location, data_file=’../data/corona_data.dat’):
    """
    Extracts COVID-19 data for a specific location.

    :param location: The name of the location (case-sensitive).
```

[1] https://github.com/CSSEGISandData/COVID-19

[2] https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

[3] https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

```
    :param data_file: Path to file holding the COVID-19 data.
                      It is expected that columns of data are
                      separated by tabs, and that there is a
                      column called "LOCATION" with names of
                      each country, region, etc.
    :return: A pandas DataFrame with COVID-19 data for the
             input location.
    """
    df = pd.read_csv(data_file, sep='\t')
    try:
        df = df[df['LOCATION'] == location]
    except:
        print(f'Could not find data for location {location}...')
        return None
    return df
```

3 Exercise 2

Part 1. As an example of how to visualize the data, we again use Afghanistan. However, note that we will not spend time on interpreting the outbreak in this country.

Before making any plots, we choose to extract the underlying NumPy arrays holding the data on time and confirmed cases:

```
time = df['ELAPSED_TIME_SINCE_OUTBREAK'].to_numpy()
confirmed = df['CONFIRMED'].to_numpy()
```

Note that this is not really necessary, because we can also pass `pandas.DataSeries` objects directly to the plotting commands

Tip 1: If you are uncertain about what kind of an object a variable `x` refers to, the command `type(x)` provides the answer!

Part 2. To visualize the data for Afghanistan, we input the arrays from the previous part to the `scatter` function provided by `matplotlib`:

```
fig, ax = plt.subplots()
ax.grid()
ax.set_xlabel('Time since initial outbreak (days)')
ax.set_ylabel('Number of confirmed cases')
ax.scatter(time, confirmed, color='black')
```

Part 3. and 4

Generalizing what we just did, we write a Python function to create plots for an *arbitrary location*:

```
def plot_confirmed_cases(location, data_file=’../data/corona_data.dat’):
    """
    Plots the number of confirmed COVID-19 cases for a specific
    location.

    :param location: The name of the location (case-sensitive).
    :param data_file: Path to file holding the COVID-19 data.
                      It is expected that columns of data are
                      separated by tabs, and that there is a
                      column called "LOCATION" with names of
                      each country, region, etc.
    :return: A matplotlib.pyplot.figure object.
    """
    # Get data
    df = get_corona_data(location, data_file)
    time = df[’ELAPSED_TIME_SINCE_OUTBREAK’].to_numpy()
    confirmed = df[’CONFIRMED’].to_numpy()

    # Make plot
    fig, ax = plt.subplots()
    ax.set_title(location)
    ax.grid()
    ax.set_xlabel(’Time since initial outbreak (days)’)
    ax.set_ylabel(’Number of confirmed cases’)
    ax.scatter(time, confirmed, color=’black’)
    return fig
```

Note that we return the figure object from the function; this allows us to customize the plot further outside of the function should we wish to do so.

Part 5. Next, we plot number of confirmed cases in Hubei and Norway:

```
fig_hubei = plot_confirmed_cases(’Hubei’)
fig_norway = plot_confirmed_cases(’Norway’)
```

To build intuition about the COVID-19 outbreak, we want to compare the data with a simple mathematical model. Therefore, we will come back to these data below, after having introduced our model of choice.

4 Using the SI-model to describe the spread of COVID-19

Most models for epidemics are some sort of compartment model [3]. These models first divide the total population into a set of boxes or compartments, each of which represents a possible "disease state" (healthy, exposed, infected, recovered, vaccinated, dead, etc.) Next, differential equations are set up to describe how individuals "flow" from one compartment to another. Obviously, compartment models come in many different flavours, and they have greatly varying complexity. Since this is an introductory project, we will use an extremely simple model, the *deterministic SI-model* [2]:

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = -\beta \cdot \frac{S(t)I(t)}{N} \tag{1}$$

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = +\beta \cdot \frac{S(t)I(t)}{N}\,. \tag{2}$$

In the above equations, N is the total population size, $S(t)$ denotes the number of susceptible people at time t, the population at risk of being infected, while $I(t)$ denotes the number of infected (sick) individuals. Since there are only two compartments, we must have $S(t) + I(t) = N$ at all times. This implies that people never die or recover from the disease, and that eventually everyone will be infected, a clear weakness with the model.

5 Exercise 3

Part 1. The analytical solution to equation (2) can be derived by first inserting $S(t) = N - I(t)$, separating variables, and integrating:

$$\int_{I_0}^{I(t)} \frac{\mathrm{d}I(t)}{I(t)(N - I(t))} \mathrm{d}I = \int_0^t \frac{\beta}{N} \mathrm{d}t = \beta t\,, \tag{3}$$

where $I_0 = I(0)$. To calculate the integral on the left, we can use use partial fractions to split the integrand into a sum of two parts:

$$\int_{I_0}^{I(t)} \left(\frac{1}{I} + \frac{1}{N - I}\right) \mathrm{d}I = [\ln I - \ln (N - I)]_{I_0}^{I(t)} \tag{4}$$

$$= \ln \left(\frac{I}{N - I}\right)_{I_0}^{I(t)} \tag{5}$$

$$= \ln \frac{I(t)}{N - I(t)} - \ln \frac{I_0}{N - I_0} \tag{6}$$

$$= \ln \frac{I(t)}{N - I(t)} + \ln \frac{S_0}{I_0}\,. \tag{7}$$

Hence, taking the exponential on both sides of the original integral:

$$\frac{I(t)}{N - I(t)} = \frac{I_0}{S_0} \cdot \exp{(\beta t)} \tag{8}$$

$$I(t) = \frac{S_0 + I_0}{1 + \frac{S_0}{I_0} \exp(-\beta t)}\,. \tag{9}$$

where $S_0 = S(0)$, and where we have used that $S_0 + I_0 = N$. Finally, the number of susceptible people are:

$$S(t) = N - I(t) = \frac{(S_0 + I_0)\frac{S_0}{I_0}\exp(-\beta t)}{1 + \frac{S_0}{I_0}\exp(-\beta t)}\,. \tag{10}$$

6 Exercise 4

Part 1. The following Python function calculates the analytical solution to the SI-model, equations (10) and (9):

```
def calc_SI_model(t, S0, I0, beta):
    """
    :param t: An array of times.
    :param S0: The initial number of susceptible people.
    :param I0: The initial number of infected people.
    :param beta: The disease transmission rate parameter.
    :return: A tuple of arrays holding S(t) and I(t).
    """
    I = (S0+I0)/(1.0 + S0*np.exp(-beta*t)/I0)
    S = S0 + I0 - I
    return S, I
```

Part 2. Next, we implement a new function that combines the SI-model with the plot function:

```
def compare_confirmed_cases_with_model(location, S0, I0, beta,
                                   data_file=’../data/corona_data.dat’):
    """
    Plots the number of confirmed COVID-19 cases for a specific
    location.

    :param location: The name of the location (case-sensitive).
    :param S0: The initial number of susceptibles in the model.
    :param I0: The initial number of infected people in the model.
    :param beta: The model disease transmission rate parameter.
    :param data_file: Path to file holding the COVID-19 data.
                      It is expected that columns of data are
                      separated by tabs, and that there is a
                      column called "LOCATION" with names of
                      each country, region, etc.
    :return: A matplotlib.pyplot.figure object.
    """
    # Calculate S(t) and I(t) from the analytical solution
    t = np.linspace(0, 250, 251)
    St, It = calc_SI_model(t, S0, I0, beta)
    # Plot the data, and return the data
    fig = plot_confirmed_cases(location, data_file)
    # Add modelled I(t) to the same figure
    ax = fig.axes[0]
    ax.plot(t, It)
    return fig
```

Part 3. The city of Wuhan is located in the province of Hubei. Since COVID-19 first arose in Wuhan [9], the data for Hubei is of special interest when studying the dynamics of the disease transmission. As remarked previously, the simple SI-model predicts that everyone in the population will be infected eventually. On the other hand, the data for Hubei indicates that the Corona virus spread very quickly after the initial outbreak, but then reached a plateau level much lower than the total population size. Assuming the data to be trustworthy, this is most likely a consequence of strong countermeasures being put in place in Hubei.

To match the model to the data, we therefore set N equal to the final number of confirmed cases in Hubei. For simplicity we also assume that at time $t = 0$, the number of infected individuals was $I_0 = 1$, hence $S_0 = N - 1$. We manually adjust S_0 and β to get a match between model and data:

```
df = get_corona_data(’Hubei’)
N = df[’CONFIRMED’].iloc[-1]
fig_hubei_model = compare_confirmed_cases_with_model(’Hubei’, N-1, 1, 0.6)
```

Part 4. For comparison purposes, it might be interesting to study how the virus spread in a more confined space. One such example comes from the cruise ship *Diamond Princess*. There were $S_0 = 712$ confirmed cases of the Corona virus aboard the cruise ship [8]. If we use the same β-value we history-matched to the Hubei data, we get:

```
compare_confirmed_cases_with_model(’Diamond Princess’, 712, 1, 0.6)
```

It should be remarked that the total number of passengers were about five times larger [8].

Part 5. As with the case of Hubei, we cannot use the total population size of Norway for S_0, because that would lead to millions of people being infected in the model calculation. A challenge with matching this particular data set is there is not a single plateau level for the disease; instead, there is a second wave starting at around 150 days. Therefore, the history-match will clearly be poorer for Norway overall than for the previous two locations.

To make things simple, we ignore the second wave and only focus on the initial period. After 150 days, the number of confirmed cases is close to 8000-9000, so we choose $S_0 = 8000$:

```
compare_confirmed_cases_with_model('Norway', 8000, 1, 0.6)
compare_confirmed_cases_with_model('Norway', 8000, 1, 0.27)
```

As seen in the above two plots, we have to lower β in order to capture the trend in the data. One interpretation is that, initially, the disease spread more slowly in Norway than in Hubei. At the same time, the countermeasures were not as effective in Norway.

7 Exercise 5

We have seen that the SI-model works quite well provided that the disease a) spreads very quickly, and b) levels out thereafter. However, there are several clear weaknessnes with the model. One big problem has to do with the fact that there is only two compartments in the model; as remarked several times, this means that people can never recover from the disease in the model, i.e., everyone will eventually be infected. A more realistic model would add extra compartments, to account for not only recovered individuals, but also dead people, vaccinated ones, etc.

Another issue has to do with the β-factor, which is a measure of the "effective probability of infection". We have so far assumed β to be constant, but in reality we know that it should change with time. While some factors could lead to a larger risk of disease transmission (e.g., mutated variants of the virus [1]), it is likely that β will decrease due to strong countermeasures put in place by the world's different governments. To capture this kind of behavior, we again use a simple model and suppose that β declines exponentially:

$$\beta(t) = \beta_0 e^{-\lambda t}. \tag{11}$$

One can prove that equations similar to (9) and (10) still hold; all we have to do is to replace the product βt with

$$\beta t \to \int_0^t \beta_0 e^{-\lambda t} dt = \frac{\beta_0}{\lambda}(1 - e^{-\lambda t}). \tag{12}$$

In the above expression, β_0 is the initial infection rate, while the exponenial decline parameter, λ, is used to capture effects of counter-measures: A high value of λ indicates strong disease-prevention, while in the limit $\lambda \to 0$ we recover our original SI-model that has no intervention effects.

Part 1. We implement the exponential decline model by modifying the functions we made previously: we include both β and λ as input arguments, and simply let $\lambda = 0$ by default:

Tip 2: If you choose this way of implementing your code, you are allowed to skip straight to the "most advanced version" in your handed-in project. This way you avoid copying and pasting almost identical code in your project, and you reduce the risk of introducing bugs!

```
# We replace the old function "calc_SI_model" by this:
def calc_SI_model(t, S0, I0, beta, lam=0.0):
    """
    :param t: An array of times.
    :param S0: The initial number of susceptible people.
    :param I0: The initial number of infected people.
    :param beta: The disease transmission rate parameter.
    :param lam: Decline parameter for exponential decline
                of beta (default: 0.0).
    :return: A tuple of arrays holding S(t) and I(t).
    """
    bt = (beta/lam)*(1.0 - np.exp(-lam*t)) if lam > 0 else beta*t
    I = (S0+I0)/(1.0 + S0*np.exp(-bt)/I0)
    S = S0 + I0 - I
    return S, I
```

Similarly, we update our last plotting function in two places by passing in the extra input argument:

```
# Replace the old "comparison function" with this:
def compare_confirmed_cases_with_model(location,
                                       S0,
                                       I0,
                                       beta,
                                       lam=0.0,
                                       data_file=’../data/corona_data.dat’):
    """
    Plots the number of confirmed COVID-19 cases for a specific
    location.

    :param location: The name of the location (case-sensitive).
    :param S0: The initial number of susceptibles in the model.
    :param I0: The initial number of infected people in the model.
    :param beta: The model disease transmission rate parameter.
    :param lam: Decline parameter for exponential decline
                of beta (default: 0.0).
    :param data_file: Path to file holding the COVID-19 data.
                      It is expected that columns of data are
                      separated by tabs, and that there is a
                      column called "LOCATION" with names of
                      each country, region, etc.
    :return: A matplotlib.pyplot.figure object.
    """
    # Calculate S(t) and I(t) from the analytical solution
    t = np.linspace(0, 250, 251)
    St, It = calc_SI_model(t, S0, I0, beta, lam)

    # Plot the data, and return the data
    fig = plot_confirmed_cases(location, data_file)
    # Add modelled I(t) to the same figure
    ax = fig.axes[0]
    ax.plot(t, It)
    return fig
```

Part 2. The actual population size of Hubei has been reported as 57.8 million [4]. Let us try using this value as input to the exponential-decline SI-model. By repeated trial-and-error, we find a combination that works reasonably well:

```
compare_confirmed_cases_with_model(’Hubei’, 57.8e6, 1, 2.0, 0.18)
```

Note that we have more than tripled the initial β-factor compared to our previous match (now $\beta_0 = 2$ vs. before 0.6) We are still able to capture the trend in the data, though, by tuning λ accordingly. This example illustrates the fact that different models may fit the same data equally well, and that values of fitted parameters do not always have a clear-cut "physical interpretation". However, as long as we apply the *same model to different locations* we may compare the fitted parameters and hopefully learn something useful.

The updated SI-model behaves differently from the original one in that the disease outbreak will not reach the whole population. However, it is still the case that the model can only account a single wave. This is because $\beta(t) \to 0$ as $t \to \infty$, which means that after some time, no one will be infected anymore, regardless of how many infectious individuals there are in the population.

Part 3. The population of Norway is approximately 5.4 million [5]. If we use the $\beta-$ and $\lambda-$values fitted to the Hubei data in order to predict the disease outbreak in Norway, we get:

```
compare_confirmed_cases_with_model(’Hubei’, 5.4e6, 1, 2.0, 0.18)
```

We see that we drastically overpredict the number of COVID-19 cases in Norway.

Part 4. By trial-and-error, we find an improved match for the initial disease transmission period in Norway:

```
compare_confirmed_cases_with_model(’Norway’, 5.4e6, 1, 0.8, 0.0898)
```

If we can assume that the value of λ represents the government response, this means that while the initial outbreak was more severe in Hubei than in Norway, countermeasures were less strict in Norway ($\beta_{0,\text{Hubei}} > \beta_{0,\text{Norway}}$, $\lambda_{\text{Norway}} < \lambda_{\text{Hubei}}$), which fits well with what has been reported in the media [7].

8 Conclusion and Discussion

What did you learn?

What are the implications of your findings? Are there any weaknesses or limitations in your approach? What can be improved?

It is surprising that the SI-model can describe the data so well. Countries are very different, but still the spread of COVID-19 are captured using similar model parameters for different places. When interpreting the data, we can therefore compare different countries quantitatively, e.g., how fast the disease spreads, and the strength of governmental responses. We find that the governmental countermeasures were much stronger in Hubei than in Norway. However, many more countries should be analyzed with the model before we can conclude that an exponential decline model for β makes sense.

The work in this report supports the notion that we can use compartment models to analyze the Corona virus outbreak. To make the model more realistic, additional compartments should be introduced, to account for people are infected without being sick (asymptomatics), people who have recovered from the virus, dead people, vaccinated people, and so on. For large countries, or for very spread-out populations, compartments could also be introduced for different regions, and then one should allow for some sort of non-neighbour flow between them to account for traffic (buses, trains, airplanes etc.)

9 Self-reflections

Self-reflections.

Self-reflections are an important part of the learning process. Take time to reflect on what you did that was good or bad. Be specific. What *specifically* were you pleased with? What did you not like? What do you want to improve next time? Do you have any suggestions or feedback about the project itself?

Aksel: I started too late on the projects, next time I will start immediately, visit the lab every Wednesday and work continuously. We worked quite well as a team. I enjoyed to see how we could use a model to analyze data, and extract knowledge. However, I think that the model is too simple to capture the complexity of how an infectious disease spreads. It would be very interesting to see how the model could be improved to capture more effects, like vaccination, social distancing, etc.

Oddbjørn: I worked on the project in-between several other tasks, and I did most of the writing work close to the deadline. Though I think the project went reasonably well, it is probably better if I start earlier next time. Being a former mathematics student, I really enjoyed learning about how the Corona virus can be modelled with differential equations. The project allowed also gave me appreciation for how difficult it can be to analyze real-world data; it would be exciting to continue in the future by gradually extending the complexity of the model to allow for more realism!

References

[1] David Adam. What scientists know about new, fast-spreading coronavirus variants. *Nature*, 2021.

[2] Antonio M. Batista, Silvio LT Souza, Kelly C. Iarosz, Alexandre CL Almeida, José D. Szezech, Enrique C. Gabrick, Michele Mugnaine, Gefferson L. dos Santos, and Iberê L. Caldas. Simulation of deterministic compartmental models for infectious diseases dynamics. *arXiv preprint arXiv:2106.02085*, 2021.

[3] William Ogilvy Kermack and Anderson G. McKendrick. A contribution to the mathematical theory of epidemics—i. *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[4] National Bureau of Statistics of China. Communiqué of the Seventh National Population Census[1] (No. 3) - Population by Region. `http://www.stats.gov.cn/english/PressRelease/202105/t20210510_1817188.html`, November 1, 2020. Accessed: 2021-08-17.

[5] Statistics Norway. Population in Norway, 2nd Quarter 2021. `https://www.ssb.no/en/befolkning/folketall/statistikk/befolkning`, Updated: 19 August 2021. Accessed: 2021-08-19.

[6] Coronaviridae Study Group of the International Committee on Taxonomy of Viruses. The species severe acute respiratory syndrome-related coronavirus: Classifying 2019-ncov and naming it sars-cov-2. *Nature microbiology*, 5(4):536, 2020.

[7] University of Oxford Blavatnik School of Government. COVID-19 GOVERNMENT RESPONSE TRACKER. `https://www.bsg.ox.ac.uk/research/research-projects/covid-19-government-response-tracker`, 2021. Accessed: 2021-08-17.

[8] Wikipedia. COVID-19 Pandemic on Diamond Princess. `https://en.wikipedia.org/wiki/COVID-19_pandemic_on_Diamond_Princess`, 2021. Accessed: 2021-08-17.

[9] Peng Zhou, Xing-Lou Yang, Xian-Guang Wang, Ben Hu, Lei Zhang, Wei Zhang, Hao-Rui Si, Yan Zhu, Bei Li, Chao-Lin Huang, et al. A pneumonia outbreak associated with a new coronavirus of probable bat origin. *Nature*, 579(7798):270–273, 2020.