Introduction

The purpose of the assignment is to simulate the spread of COVID-19 in some specified countries, using a set of data that are provided to you. For this matter, you need to create a sample population of people according to the data provided (sample ration is provided in the `test.py`, for example one sample per one million population). This should be done according to the age structure in each country (percentage of population in each age group).

Then for each individual in the sample you run a simulation (a Markov Chain similar to assignment 2) for a specific period of steps (days). Starting and ending dates are provided to you in (the `test.py`).The Markov chain simulation should consider 5 states (H,I,S,M,D) which are described later. This result needs to be exported to a CSV file (`a3-covid-simulated-timeseries.csv`).

At the end you need to accumulate data for each specified country and each date (see the Output section). The result should be saved as (`a3-covid-summary-timeseries.csv`). Then you need to call `create_plot()` function, provided to you ,to create a chart (`a3-covid-simulation.png`)

About Creating Samples

You need to create samples for each specified country, according to the *sample ratio*, the population of the specified countries, and the age structure (age group) distribution in those countries. For example if the specified countries are 'Australia' and 'Sweden'. Then by reading `a3-countries.csv` you notice that Australia has a population of 25921089 which means for a sample ratio of 1e6 you need to create 25 samples. However, you also notice from the same file that the age group distribution in Australia is like this: less_5 ⇒ 6.5% ,5_to_14 ⇒ 12.7% ,15_to_24 ⇒ 12.2%, 25_to_64 ⇒ 52.3% and over_65 ⇒ 16.2%. Which means that the number of samples for each age group should be 25*percentage/100 (rounded to integer). With this approach, some age groups might have zero sample. Also, note that you should do the same for other specified countries (Sweden in this example) and keep them along other countries.

You can imagine that a smaller sample ration can create finer simulation results. However, we tried to keep the value in a range that does not take a long time for simulation. You may want to test your code, for your own sake, with smaller values.

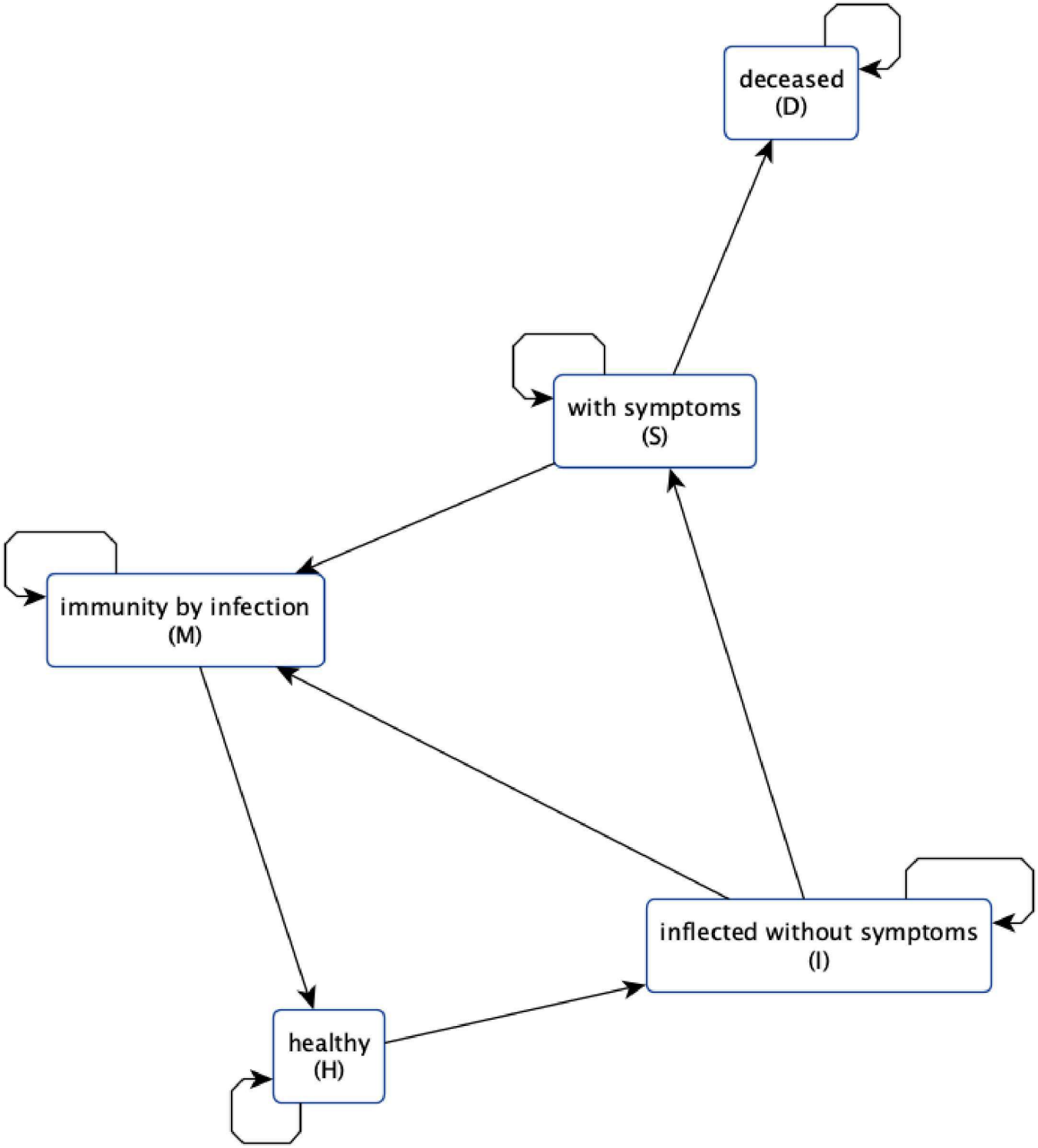
About Time-Series

There is a starting date ('2021-04-01' in the `test.py`) and ending date ('2022-04-30' in the `test.py`) for the simulation (395 days totally). You need to create a data structure to store the state of each sample in each date in this range (you may use standard features such as list and dictionary, or instead use `DataFrame` in `pandas`). This data structure is going to be filled in by your simulation result. Please note that starting and ending dates might change in the evaluation (so the total number of days in the simulation). This time-series data structure is going to be save to a CSV file later.

Note about the size of the time-series. For an example, for countries Afghanistan, Sweden, and Japan with 166 million population totally, with a sample ratio of 1e6 you will have 166 sample population. Then for 395 days that will be 166*395 = 65965 records to store. There is a technical suggestion for implementing this time-series, later in this instruction.

About Simulation for Individual Samples

The simulation of infection with COVID-19 for each individual is shown in the Markov Chain in the image below:



This specific state machine is technically a form of continuous-time Markov chain (CTMC). However, for the sake of implementing this assignment, you do not need to know about Markov chains (or more specifically about CTMCs). Nevertheless, it is a good chance to read about Markov chain concept, since it is instrumental in many real-world scenarios.

In this Markov chain, an individual begins in the state 'healthy (H)' and by some *transition probability* it changes state to 'infected without symptoms (I)', 'with symptoms (S)', 'deceased (D)', 'immunity by

infection (M)' or even back to 'healthy (H)'.

There is also a *holding time* for each state, which means how many days a person should stay in a state before a possible move to another state. Step of this simulation is in *day*. A holding time of 0 is considered the same as holding time 1 (staying one day in that state before trying the transition probabilities to change state)

As you can see no *transition probability* value is specified in the image, since the transition probabilities are specific to each age group and provided in the `sim_parameters.py`. That means depending on whether the simulated person belong to which age group the values of transition probabilities can be different.

Please, refer to assignment 2 instruction, if you require some more explanation about implementing a Markov chain.

Team Setup

For this assignment you need to form a team of max 3 students. Beside dividing tasks, it is a good opportunity to practice programming in teams, as it is in the real-world situation. Please note that all members of the team are required to know all pieces of the code, and attend the final presentation session. You may want to learn more about 'pair programming' practice and apply it when developing this assignment.

Learning Outcomes

In this assignment, a series of learning outcomes are considered:

1. Implementation of this assignment involves many features of Python programming.
2. There is a focus on working with data. After, finishing this assignment successfully you may gain confidence in implementing various types of data processing related projects (which is a base for other trending topics, such as machine learning).
3. In real world situation, you always need to search and find techniques, technologies, and solutions beyond you already have learn. This assignment is similar in that sense.
4. This ia real world problem (i.e. spread of COVID-19 infection), and beside your programming skills you need to understand the problem (as it is the case in real world programming situations)
5. Learning the skill (and challenges) of working in a team is another outcome.

Environment Setup

The name of assignment file should be `assignment3.py`. You have the possibility of splitting code amongst various files, however, the file `assignment3.py` should exist with a `run()` function that will be described later.

You need to install `pandas`, and `matplotlib` libraries on your computer (or specifically in Python's environment your work with). Check yourself if anything else should be installed.

You need to download these two files in your working directory: . `helper.py` which provides visualization (creating the output chart) functionality for your final output. You need to import it in your `assignment3.py`. . `sim_parameters.py` which you import in your file and it will provide two set of parameters in dictionary type (will be discussed later)

Libraries You Can Use

You can use `numpy`, `pandas`, `functools`, `matplotlib`, `unittest`, `pathlib`, `os`, `random`, `doctest` and `csv` libraries (or any other library/module that does not need installation). Please you do not need them all, that is just a kind of flexibility for you.

This assignment is considered to be implementable using standard features of Python, such as lists and dictionaries. However, we do not prevent you using any of the above mentioned libraries (specially `pandas`). In real life, it is more practical to use such libraries (such as `pandas` in our case). If you are interested in using `pandas` I will provide you with some slides and explanation for some features in `pandas` that can help in this assignment. Otherwise, you can use the standard features of Python.

Input Data

There is one input dataset, in CSV form, that you need to read (`a3-countries.csv`). Also two dictionaries are provided to you in `sim_parameters.py` file which you need to to import (using `import` keyword). Four parameters are passed to a `run()` function that you implement in your `assignment3.py` file. This means your `run()` function should have these specific input arguments: `countries_csv_name`, `countries`, `start_date`, `end_date`, `sample_ratio`.

The `countries_csv_name` argument is name of CSV file you need to read (however, it is fixed in the `test.py` and that is `a3-countries.csv`) The file holds demographic and age-structures (**age groups**) information about 153 countries. As usual, the first row of the CSV file is the header, containing column names. The actual data is retrieved from https://github.com/owid/covid-19-data/tree/master/public/data (https://github.com/owid/covid-19-data/tree/master/public/data) https://ourworldindata.org/coronavirus#coronavirus-country-profiles (https://ourworldindata.org/coronavirus#coronavirus-country-profiles) but got curated and filtered to make it simpler for this assignment.

In `a3-countries.csv` each row corresponds to a country. The description of columns are as follows:

1. `country`: the name of the country
2. `population`: the population of the country
3. `median_age`: the median of the age in the country

4. `less_5`: the percentage of people aged less than 5 years old
5. `5_to_14`: the percentage of people aged 5 to 14 years old
6. `15_to_24`: the percentage of people aged 15 to 24 years old
7. `25_to_64`: the percentage of people aged 25 to 64 years old
8. `over_65`: the percentage of people aged 65 or more years old

In `sim_parameters.py` two dictionaries are defined. One for transition probabilities between states (`TRASITION_PROBS`), and one (`HOLDING_TIMES`) for holding times. Please note that each dictionary is two levels nested, since it required to specify parameters for each age group.

```
TRASITION_PROBS = {
    'less_5':
{
    'H':{'H':0.7, 'I':0.3},
    'I':{'I':0, 'S':0.5, 'M':0.5},
    'S':{'S':0, 'D':0.1, 'M':0.9},
    'D':{'D':1},
    'M':{'M':0, 'H':1}
},
...
}
```

```
HOLDING_TIMES = {
    'less_5':

    {'H':0, 'I':4, 'S':14, 'D':0, 'M':120}

,
...
}
```

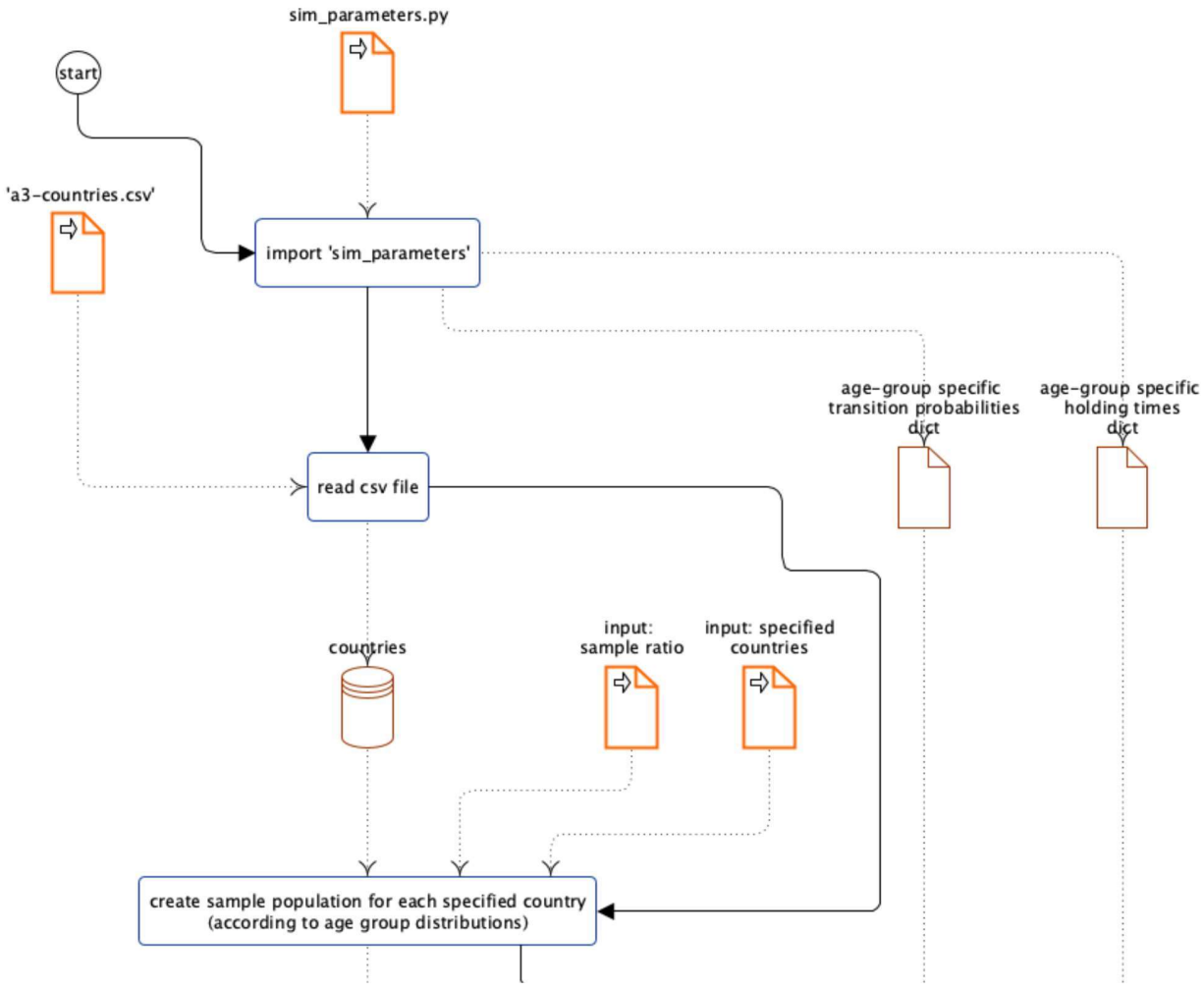
Program Structure and Flow

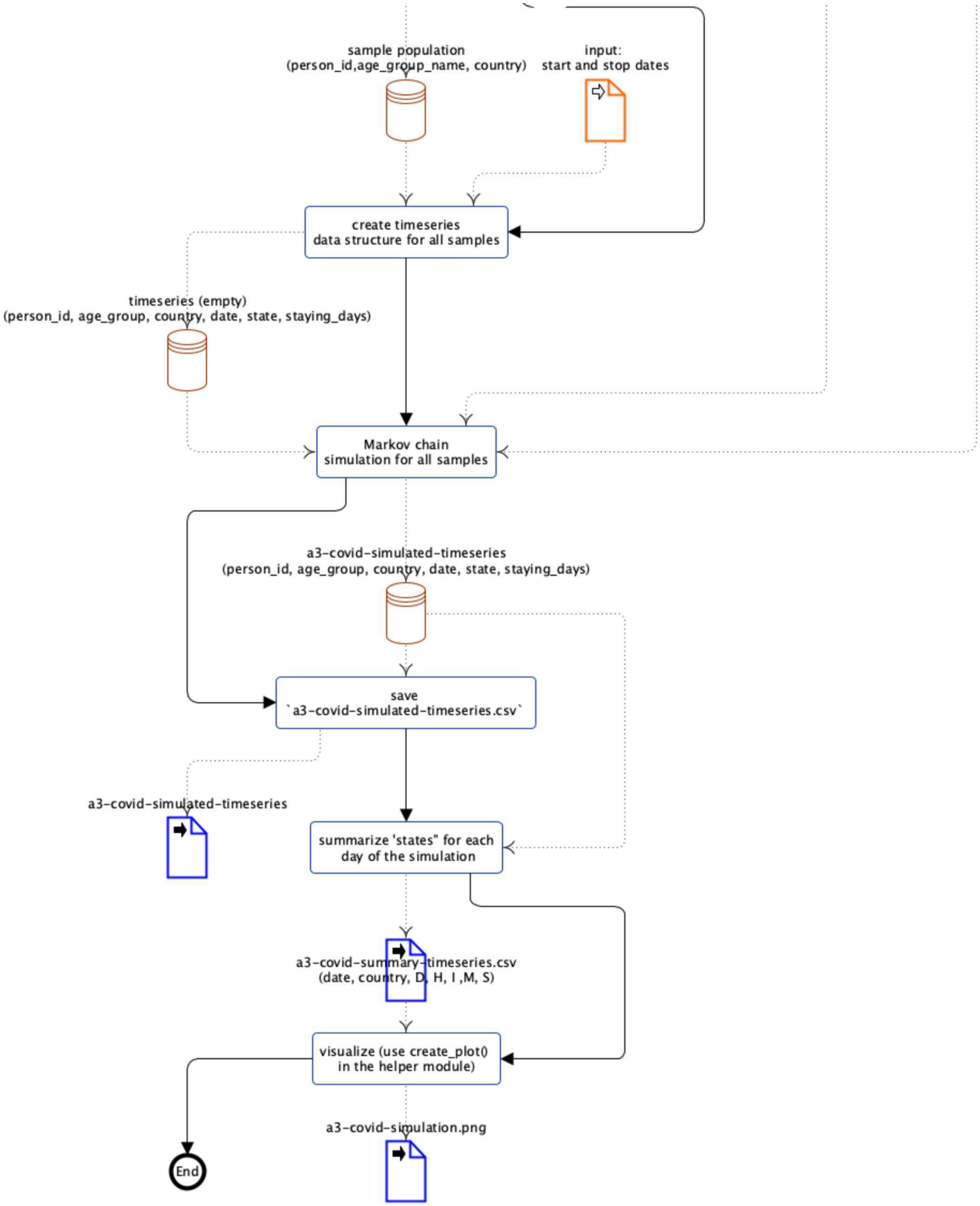
In this assignment we do not impose the flow of the program but focus on the final outputs, however you might find it helpful if you try the following flow:

1. Read the CSV file and store it in a data structure.
    a. If you use standard features of python that could be a list of dictionaries (especially if you `use csv.DictReader()`).
    b. If you use `pandas` you can use its CSV reader to get a `DataFrame` (this is the recommended way).
    c. If you use `numpy` you can use `array` class (also a good way).
2. Create a sample population according to the population of each specified country and its age group distributions.
3. Create a timeline for all individuals in the population and for the specified days of simulation

a. Important: You have two ways for this. In the first one you can imagine a two dimensional list/dictionary (list of dictionary or dictionary of lists) where each row is for an individual and each column represent a date (A dictionary of individuals, where key has a list (a time-line) value). So, there will be as many rows as the number of individuals and as many columns as the total number of simulation steps (days in our case). This is usually called a *wide table*. However, there is a another approach, which is called *long table* and is usually more recommended. In long tables, there is a limited number of columns and instead for each individual at each date you create a row. Hence, the number of rows will be number-of-individuals multiplied by number-of-simulation-steps. The *long table* approach is usually easier to handle (subseting, filtering, mapping, reducing, …) in compare to *wide table*. Look at the output CSV example (in this instruction) which is an example of a wide-table.

4. For each individual in the timeline run the Markov Chain specified from start to the end of dates. Fill in your timeline data structure. Save it in `a3-covid-simulated-timeseries.csv`.
5. Summarize (accumulate) the number of states for each date for each specified country. An example of the required format is presented in the 'Output⇒The Chart' section. Save it in `a3-covid-summary-timeseries.csv`
6. Call `create_plot()` function to plot the final result, which result in `a3-covid-simulation.png`





Output

There are three outputs for this assignment: a CSV file (`a3-covid-simulated-timeseries.csv`), a CSV file (`a3-covid-summary-timeseries.csv`) and a chart (`a3-covid-simulation.png`). You create the CSV files as

specified below, but for the output chart you just need to call the `create_plot()` function which is provided to you in the `helper.py` module. Since the chart is created automatically for you.

The CSV Files

The simulation CSV output should be a file called `a3-covid-simulated-timeseries.csv`. This a csv file with these columns names (the first row): . `person_id`: the id of an individual in the simulation . `age_group_name`: which age group that person belongs . `country`: which country that person belongs . `date`: what is the date of this information (i.e. date of the state) . `state`: in which state was the person (on that specific date) . `staying_days`: how many days the person has been in this state .. for the state of the first date the previous state is health (H) .. Use only one letter (H, I, S, D, or M) for indicating the state

Look at this example:

```
id,age_group_name,country,date,state,staying_days,prev_state
0,less_5,Afghanistan,2020-04-01,H,0,H
0,less_5,Afghanistan,2020-04-02,H,0,H
0,less_5,Afghanistan,2020-04-03,H,0,H
0,less_5,Afghanistan,2020-04-04,H,0,H
0,less_5,Afghanistan,2020-04-05,H,0,H
0,less_5,Afghanistan,2020-04-06,H,0,H
0,less_5,Afghanistan,2020-04-07,H,0,H
0,less_5,Afghanistan,2020-04-08,H,0,H
```

The second CSV file, is the summary of state for each date and each country (should be named `a3-covid-summary-timeseries.csv`). This a csv file with these columns names (the first row):

1. `date` the date in the simulation
2. `country` which country (in that specific date)
3. `D` number of deceased cases
4. `H` number of healthy cases
5. `I` number infected without symptoms cases
6. `S` number infected with symptoms cases
7. `M` number of cases with immunity (because of previous infection)

Look at this example:

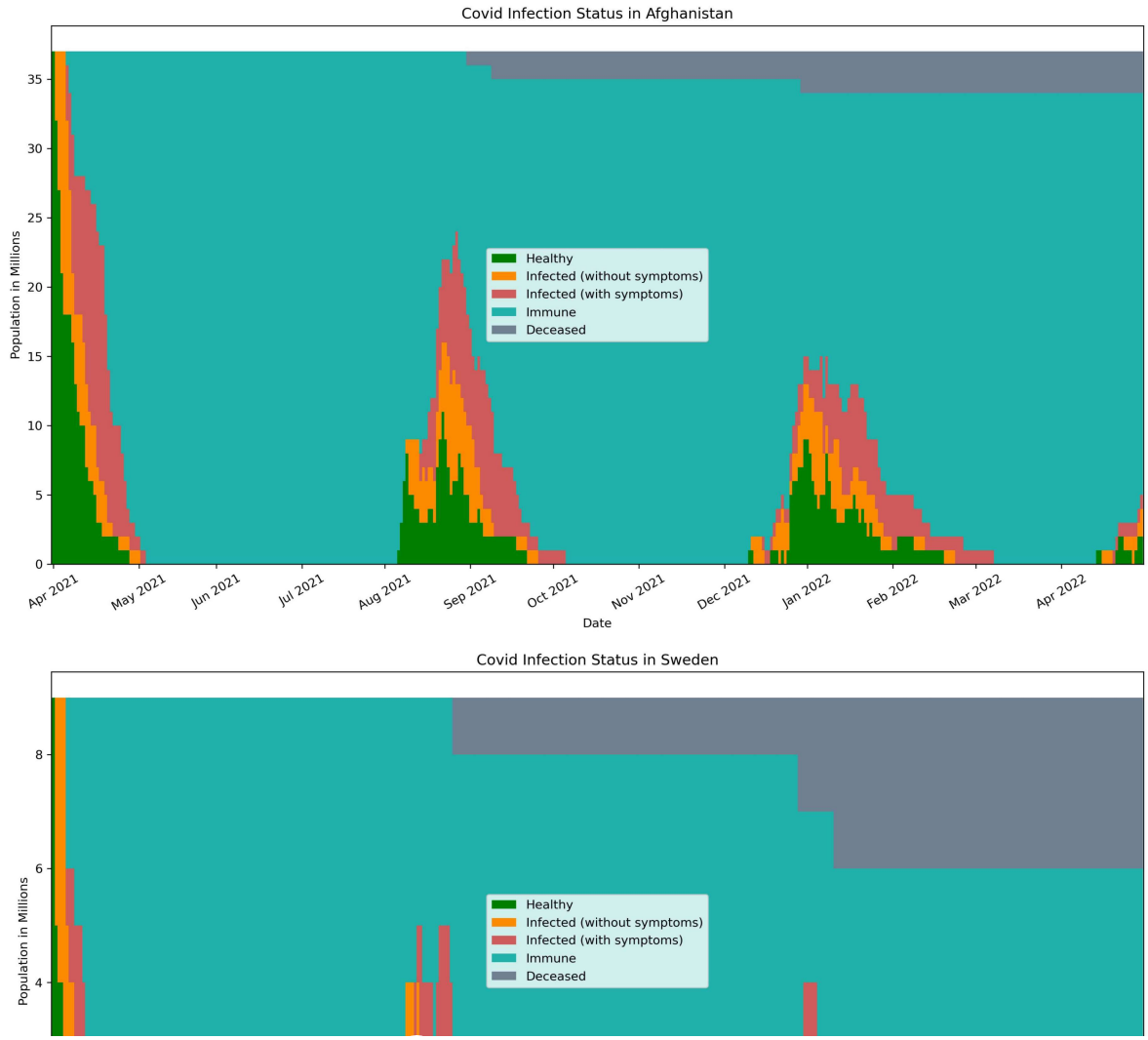
```
date,country,D,H,I,M,S
2021-04-01,Afghanistan,0,37,0,0,0
2021-04-01,Japan,0,121,0,0,0
2021-04-01,Sweden,0,9,0,0,0
2021-04-02,Afghanistan,0,26,11,0,0
2021-04-02,Japan,0,95,26,0,0
```

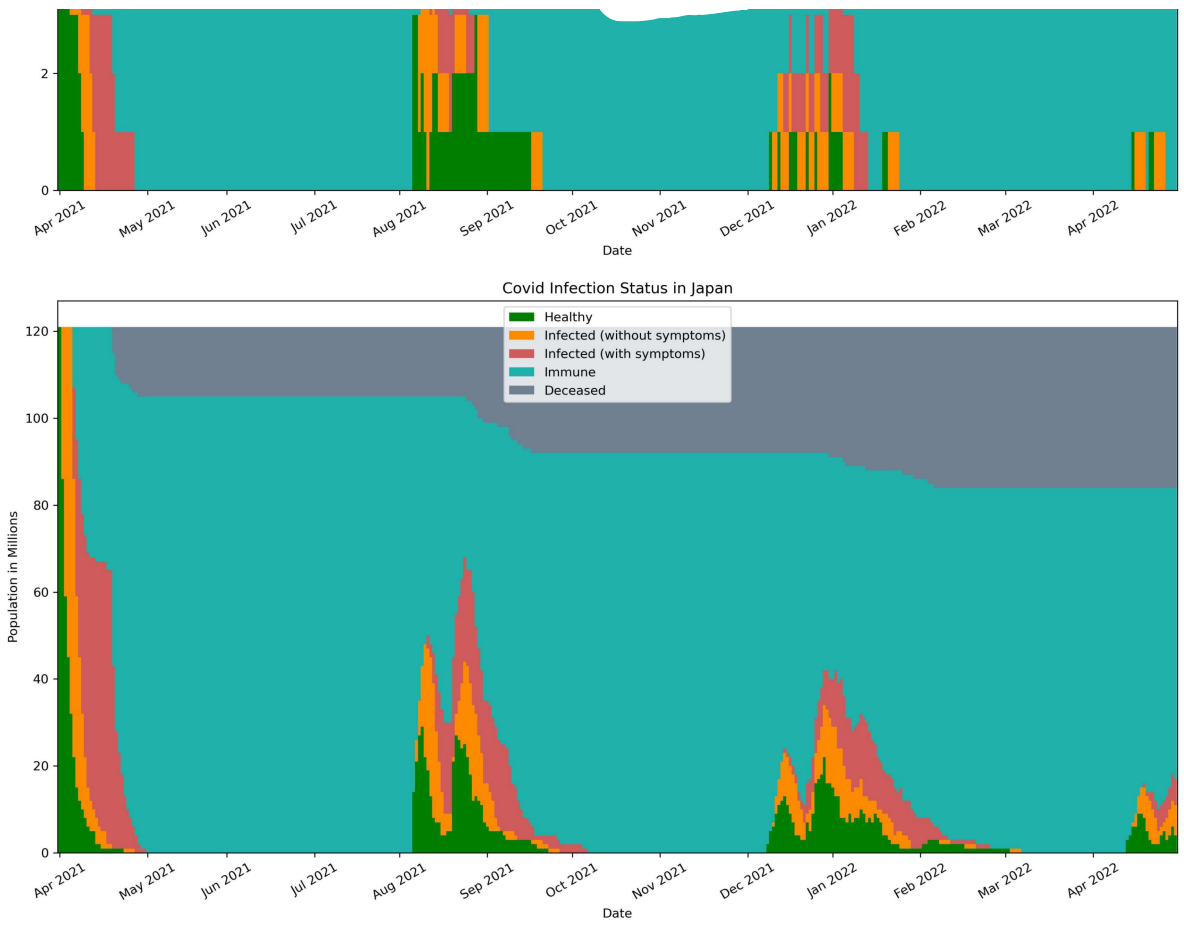
The Chart

You need to call the `create_plot()` (in the `helper.py`) after creating the `a3-covid-summary-timeseries.csv` file.

The generated chart should look something like below image (for that specified countries). Please note that due the random nature of the simulation (and that we do not use `seed()`) your result might be different each time of running. At the same time, please note that in the evaluation of your code, we might choose other countries to see the result. You are encouraged to play with your code and test different countries.

Note the waves of COVID-19 infection in this chart. Also note how different states (colors) appear after each other. These are signs that your simulation has possibly worked correctly.





Submission Instructions

1. ONLY submit your file when your code passes the test.py (creating the two CSV files and the plot, with no error)
2. Follow exactly the naming for `run()` function and its arguments
3. For all the Python files you upload, you need to upload both the python file (such as `assignment3.py`) and a file of the same content with `.txt` extension (that is `assignment3.txt`).
4. Upload all the output files, that is `a3-covid-simulated-timeseries.csv`, `3-covid-summary-timeseries.csv` and `a3-covid-simulation.png` file also.
5. Upload a zip files that contains everything (all python files, all output CSV files and the plot PNG file) .
6. Care about avoiding plagiarism situation

7. Again note that each member of the team should be able to explain every piece of the code, and attend the (possible) presentation session.

Some Scientific Notes (not a part of the assignment, you may skip reading this section)

Related Topics

The simulation you implement in this assignment is closely related to few trending topics. One of them is *digital twins* where a digital replica is created for each entity (persons or devices) in a setting, all parameters about them are gathered in real-time, and a predictive model about their future situation is created (and kept updated) continuously. Other wordings that you might find related are *agent based modeling* and *micro-simulations*.

Also, while this assignment does not involve any kind of *statistical learning* (i.e. machine learning), but one might think of methods which are about forecasting time-series such as autoregressive models (AR) or long short-term memory (LSTM) models. Skill gained from this assignment can be a base upon which those models get implemented.

Limitations

Although this simulation demonstrates the wave nature of epidemies, but some parameters are removed from the problem to make it easier for you to implement. However, those parameters are important in real world cases. For example:

1. In this simulation countries are considered similar in their efforts in control of the spread of COVID-19 and in their success in vaccination. However, we know that is not the case in reality. The difference between countries in these two factors might be reflected in the probability of transition from H (healthy) to I (infected without symptoms) states in our simulation, which should be both country-specific and date-specific, while we considered it fixed (for each age group).
2. Also, the transition probability from S (infected with symptoms) to D (deceased) is again both country-specific and date-specific, regarding the capacity and quality of health systems in each country, while still we considered it fixed for all countries, all dates.
3. In reality, there is no fixed holding time for each state, but it is a distribution of probabilities, but we considered it fixed.
4. Provided infection rates (from H to I) are somehow exaggerated, and it happens in much less rates. However, high rates creates better visualization.

And there lots of more subtle (but important) considerations to consider.