User Manual

December 22, 2020

1 Introduction

The following is a manual for usage and extension of agent-based modeling software for high-granularity simulations of COVID-19. The details on the model, software, its functionality, and accompanying database can be found in our manuscript and its supplementary material. This manual instead gives a more general overview of the code and its usage. This manual is under development and will be progressively improved for the next couple months. For basic information the user is also referred to the README.md file at the root of the repository.

The software is fairly efficient, as documented in the later part of this manual. It is serial, and requires modest amount of RAM (less than 1 GB for a population of 80,000 with heavy data collection). The software is functional on MacOS and Linux, it most likely will not work as is on Windows. It requires Python 3.X and C++ compilers with the C++11 standard. Simulation wrappers and post-processing are written in MATLAB, but those scripts can readily be replaced by Python 3.X, which is planned for the future.

The code is entirely open-source and questions, feedback, as well as collaboration are highly encouraged. To contact, email us or create a new GitHub issue. The email contacts are:

Agnieszka Truszkowska (author): at4584@nyu.edu

Dynamic Systems Laboratory (organization):

1.1 GitHub repository

The repository consists of the following elements at the top (root) level:

- `include` - directory with all the header files
- `src` - directory with all the C++ source files
- `scripts` - Python scripts for use throughout the repository
- `tests` - all tests developed for this code
- `parameters` - data and scripts for fitting of some of the parameters
- `simulations` - simulations, specifically,
  - `NewRochelle_population` - model validation and optimization of testing practices
  - `vaccination_study` - vaccination strategies without restrictions
  - `code_complexity` - code for generating data for complexity estimates.

  Specific features of these simulation sets are available in their own `README.md` files.

2 Code structure overview

The code consists of several groups of source code, linked by a user interface, located in the header 'abm.h'. It is recommended that the interface provides interactions with all features that are supposed to be high level and accessible by a user. These include various data collection and functionality such as vaccination. This section describes the structure and components of each group of code that is currently part of the software.

2.1 ABM class

This class represents the user interface. It has all the components needed for running a simulation, as described in the header file and the documentation. Specifically, it consists of the following functionality:

1. Loading the buildings in town, initializing households, schools, retirement homes, hospitals, and general workplaces.
2. Loading the agents, selecting initially infected agents. There are two ways to initialize COVID-19. One is by randomly selecting a fixed number of agents to have COVID-19, second to read it from the file. The user chooses the method in the initialization step.
3. Driving the progression of the epidemic.
4. Implementing non-pharmaceutical interventions, time-dependent testing prevalence, vaccination (random and selective) and accounting for presence of the symptomatic agents that do not have COVID-19.
5. Data collection, including returning copies or read-only references to all the agents, locations, and various infection-related quantities.
6. Printing information on agents and locations.

2.2 Various simulation classes

Following lists classes related to various elements of the model, what they consist of, and their basic functionality.

- Agent class
  This class contains all the attributes of an agent, including properties like age and states such as whether the agent is a hospital employee, is infected etc. All states are accompanied with appropriate setters and getters where applicable. The class also stores information on various times related to the epidemic, e.g. time when an agent's latency period ends, time of hospitalization, recovery and others.
- Infection class
  Currently contains all the stochastic functions and provides interface for calling them. It also starts and maintains the class `rng.h` which is a random generator wrapper. The design is not perfect, but intended to have only one generator, seeded at the begining of the simulation.

- Flu class
  Interface for all the functionality related to agents that have COVID-19-like symptoms, but not COVID-19 (i.e. they, for instance, have influenza). This class manages testing and test results of those agents (high level - low level settings are from `tesing.h`), adding, removing, and swapping them, as well as setting when to test and put under home isolation.

- Testing class
  Manages time depending testing prevalence. Loads time dependence, then at each step, as prompted by the ABM class, checks if there is a change in testing percentages. If there is, calculates testing prevalence for all considered agent categories, including agents with flu.

- Contributions class
  Functionality for computing infection contributions for various locations.

- Utils class
  Utility functions - floating point comparison and a function for transforming all string letters to lowercase.

2.3 Transitions classes

These classes manage all the state changes of the agents during the epidemic. `transitions.h` is the interface called by `abm.h`. The rest of the classes are written for specific types of agents as indicated by their names. Classes manage the possibility an agent will get infected and anything that happens in that case, including testing, treatment, and removal.

2.4 States manager

These classes set the states of an agent that accompany their status changes (e.g. when an agent is supposed to be tested, the class will set all the right flags to true or false). There are two types of classes, `regular_states_manager.h` contains all the management of regular agents and agents with COVID-19 like symptoms due to other diseases. `hsp_employee_states_manager.h` governs changes for hospital employees and patients that were admitted with a condition other than COVID-19. `states_manager.h` is a class that provides example interface for new cases and is not currently used.

2.5 Places

This represents an inheritance hierarchy with `place.h` being a base class and a generic place. Other places are all derived from it right now, with changes where applicable.

2.6 IO operations

There are three main classes in this group:

- FileHandler.h
  This class literally manages the files at a low level, and performs checking of success of IO operations. It is recommended to use this class instead of direct file access.

- abm_io.h
  Interface for reading and saving data, including data arrays of various types. Recommended instead of manual saving.

- load_parameters.h
  Functionality for loading the parameter list as a map and loading the age-dependent quantities.

3 Testing

All the functionality in the code is accompanied by extensive test suite. The tests can be ran automatically using Python driver scripts provided in each testing category. They can also be ran all at once with the script `run_all_tests.py` located in the root of `tests` directory. The tests were confirmed to work on MacOS and Linux operating systems. They are color coded: a passed test is indicated as a different color than a failed one, alongside with proper verbal indication and more detailed failure message. When ran together, the tests take up to an hour to run.

4 Usage

The user is referred to the Examples section until this section is developed.

5 Examples

The code is accompanied by several examples published as part of our manuscript. The examples are actual studies in the manuscript and are complete - the code, compilation, and inputs are operational and will reproduce the results from the manuscript. The setup requires a compilation that is done by running `python3.X compilation.py`. After the compilation the examples can be ran with MATLAB wrappers termed `parametric_driver.m`, or equivalent in case of vaccinations.

6 Extending the code

The user is encouraged to extend the code, contact us, and/or raise GitHub issues.

6.1 Terminology

The user is referred to our manuscript until this section is developed.

6.2 How to cite

If you find our software useful, kindly cite both the GitHub repository and the manuscript that describes it as: