A C++ implementation of the SIR model and beyond

Simone Coli   Giuseppe Sguera

Abstract

Within this project we implement a program that uses the SIR model to simulate the outbreak of a pandemy. The program can olso operate backwards and estimate the SIR parameters of a given set of data by using the least squares method.

As variations on the theme we implement a forecasting program based on the logistic model, as well as a real-time graphic simulation of the spreading of a flu among a closed group of people.

All the code is written in C++.

External libraries needed to properly run the program (some of them must be installed onto the user's computer):

- Lyra;
- SFML;
- Doctest;
- CMake (optional);

The work is divided among three directories:

mandatory_SIR_model It contains the headers and source code of the main program.

logistic_model It contains the headers and the source code of the logistic model program.

graphic_simulation It contains the headers and the source code of the graphic simulation program.

More about the SIR model at [1], [2], [3].

Contents

1 Mandatory SIR model

The code is spread over many translation units.

Classes are defined in headers, methods and free functions are declared in headers and defined in the source code. The file `pandemy.test.cpp` contains tests source code while `root_pandemy.C` is a root macro which can help to graphically visualize the program output.

To compile the program is suggested the use of CMake:

```
$ cmake -S . -B build/
$ make sir_model    # the program executable
$ make sir_model.t # the tests executable
```

otherwise you can use your favourite compiler. For example:

```
# to generate the program executable
$ g++ State.cpp Virus.cpp Pandemy.cpp Least_Squares.cpp \
Print.cpp Parser.cpp main.cpp -o sir_model
# to generate the tests executable
$ g++ State.cpp Virus.cpp Pandemy.cpp Least_Squares.cpp \
pandemy.test.cpp -o sir_model.t
```

Using CMake will compile with `-Wall -Wextra -fsanitize=address` options enabled.

1.1 Purpose

There are two modes the program can work in: the sir simulation mode and the fitting mode.

While running in the sir simulation mode, given the initial state of a population (number of susceptibles, infected and removed) and the parameters of the pandemic (beta and gamma values), the program simulates the developing of the epidemy by numerically resolving a system of three ODEs and prints on standard output the day-by-day count of susceptibles, infected and removed from day 1 until the day choosen by the user.

While running in the fitting mode, given a file containing the day-by-day count of susceptibles, infected and removed, the program estimates the parameters of file pandemic by means of the least squares method and prints them on standard output.

1.2 Running the program

First you must choose whether to run the sir simulation mode or to run the fitting mode. The two choices are mutually exclusives. If no choice is made, the program will do nothing.

```
$ ./sir_model sir # runs the sir simulation mode
$ ./sir_model fit #runs the fitting mode
$ ./sir_model.t #executes tests
```

The other options (such as the initial state of the population or the file containing the data to fit) are entered from command line by using the specific token.

For more detailed instructions use the help command:

```
$ ./sir_model --help # general help
$ ./sir_model sir --help # sir simulation mode help
$ ./sir_model fit --help # fitting mode help
```

If gnuplot is installed, it is possible to use it to graphically visualize the program output. Alternatively, if ROOT is installed, it is possible to load the macro and execute it.

```
$ ./sir_model sir -b 0.8 -g 0.3 -S 900 -I 100 -R 0 -d 30 > dati.dat
$ gnuplot

        G N U P L O T
        Version 5.2 patchlevel 8    last modified 2019-12-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2019
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'qt'
gnuplot> plot "dati.dat" u 1:2 w l, "dati.dat" u 1:3 w l, \
 "dati.dat" u 1:4 w l
```

or

```
$ ./sir_model sir -b 0.8 -g 0.3 -S 900 -I 100 -R 0 -d 30 > dati.dat
$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.22/00                        https://root.cern |
  | (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Jun 14 2020, 15:54:05                   |
  | From tags/v6-22-00@v6-22-00                                        |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'        |
   ------------------------------------------------------------------

root [0] .L root_pandemy.C
root [1] pandemy("dati.dat")
```

1.3 Classes and methods

The main classes of the program are:

Pandemy The class core of the program. It contains all the data of the pandemic to simulate (the state and the parameters). An object of type *Pandemy* it's initialized by giving it as arguments a *State* object and a *Virus* object (see below).

Methods of the class:

- **progression** It takes as argument the duration in days of the simulation and returns a vector containing the day-by-day count of susceptibles, infected and removed.
- **get_data** It takes as argument the name of the file containing the data to fit with the fitting mode and returns a vector containing the data acquired.
- **get_state get_virus** They respectively return the *State* and the *Virus* objects *Pandemy* has been initialized with. This methods are not used in the main program, they are needed only for testing.

Least_Squares The class that deals with the fitting part. By taking as argument of initialization a vector containing the data to fit, its member functions implement the least squares method (from now on LSM) and the compute of the chi square.

Methods of the class:

- **Chi_Square** It returns the chi square of set of data relatively to a theoretical prevision.
- **get_parameters** After appyling the LSM, it returns a *Virus* object initialized with the estimated parameters of the data fitted. The decimal precision of the fitting is given as argument of the method.

Minor data structures - but not less important - are:

State A simple struct to store all the information of the state of the population, such as the number of people belonging to each compartment.

`i_sigma` is a value used to compute the chi square and represents the uncertainty associated with the number of infected.

Virus Same a the *State* struct, it contains the information of the disease causing the pandemic, that are the transmission rate (β value) and the inverse of the infection period (γ value).

Parser

For both the *State* and *Virus* struct is defined the `operator==`, which turns useful for testing purposes.

Not all the code is gathered in data structures and classes, there are some free functions too. The choice not to define another class is due to their marginal role in the program, they are not related, in fact, with the SIR model nor with the LSM. Those free functions are:

copy_round A simple but effective function to round the output of the progression meth

print (disabled)

print_simple

1.4 Limits of the program

2 Logistic model

2.1 Purpose

2.2 Running the program

2.3 The classes and their methods

2.4 Limits of the program

References

[1] https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology.

[2] Gaeta, Giuseppe (2009), *Modelli Matematici in Biologia*, Springer.

[3] E. Sadurní, G. Luna-Acosta (2021), *Exactly solvable SIR models, their extension and their application to sensitive pandemic forecasting*, Springer Nature.