Epidemiological Agent-Based Modelling Software (Epiabm)

KIT GALLAGHER** (iD)

IOANA BOUROS** (iD)

NICHOLAS FAN**

ELIZABETH HAYMAN** (iD)

LUKE HEIRENE** (iD)

PATRICIA LAMIRANDE** (iD)

ANNABELLE LEMENUEL-DIOT (iD)

BEN LAMBERT (iD)

DAVID GAVAGHAN (iD)

RICHARD CRESWELL (iD)

*Author affiliations can be found in the back matter of this article

**These authors contributed equally to this research

]u[ubiquity press

ABSTRACT

Epiabm is an open-source software package for epidemiological agent-based modelling, re-implementing the well-known CovidSim model from the MRC Centre for Global Infectious Disease Analysis at Imperial College London. It has been developed as part of the first-year training programme in the EPSRC SABS:R³ Centre for Doctoral Training at the University of Oxford. The model builds an age-stratified, spatially heterogeneous population and offers a modular approach to configure and run epidemic scenarios, allowing for a broad scope of investigative and comparative studies. Two simulation backends are provided: a pedagogical Python backend (with full functionality) and a high performance C++ backend for use with larger population simulations. Both are highly modular, and include a comprehensive testing suite (including 100% coverage on unit tests alongside a range of functional and integration tests), as well as complete online documentation for ease of understanding and extensibility. Epiabm is publicly available through GitHub at github.com/SABS-R3-Epidemiology/epiabm.

CORRESPONDING AUTHOR:
Kit Gallagher

Mathematical Institute, University of Oxford, UK

kit.gallagher@maths.ox.ac.uk

(1) OVERVIEW

INTRODUCTION

As a result of the recent COVID-19 pandemic, a wealth of epidemiological models have been developed, or adapted from pre-existing models. These range from simple, population-wide models which predict the total number of infected individuals in a population [10], to models with spatial variation/age stratification to track the case rates among different groups within a population [17, 13]. Some of the most complex models, often used to advise government and policy-maker decisions, are agent-based models (ABMs) [11]. The representation of populations on an individual scale by ABMs allows for the modelling of complex spatial and behavioural phenomena which are not straightforward to account for in traditional population-averaged, compartmental differential equations models. In these spatially homogeneous models, only the total number of individuals at each step of disease progression (e.g. Susceptible, Infected, Recovered) is modelled. In contrast, implementing individual agents allows for complex inter-agent interaction networks with more realistic transmission dynamics, where interaction probabilities can be affected by each individual's spatial position and age.

One of the most influential models in the UK during the COVID-19 pandemic has been the CovidSim model [8], developed by the MRC Centre for Global Infectious Disease Analysis at Imperial College London. The model was initially designed to support influenza pandemic planning [9]. At the start of the COVID-19 pandemic, the code was very rapidly adapted to enable modelling of the initial stages of the outbreak. Notably, the model was used to produce the high profile 'Report 9', which considered the impact of various non-pharmaceutical interventions (NPIs) on the transmission of COVID-19 [7], and this report is widely held to have been influential in the UK government's decision making [1]. This model has a higher level of spatial and behavioural complexity than alternative age-stratified models [2, 4, 12, 5].

CovidSim is highly efficient, allowing large-scale simulations to be run. The model and code were originally created to underpin a time-critical academic publication. Because of this, the codebase is not (and was not intended to be) easily extensible. A range of changes to the code and software engineering practices have therefore been suggested in the academic literature, including making it more modularised; controlling the random number seeding to make results entirely reproducible; and including comprehensive code testing and documentation [14, 16, 6].

The development of a fully modular and well-documented version of CovidSim code would therefore have multiple potential benefits both for modelling and responding to future pandemics, and as a pedagogical tool. The modularity will enable researchers across the epidemiological modelling community to isolate and characterise dominant transmission mechanisms and viral characteristics in agent-based models, and allow flexible configuration of interventions. In this paper, we present such a modular and fully documented re-implementation of the CovidSim model, which we call *Epiabm*, that adheres to professional software development principles [3, 15], for use in research and education settings. The Epiabm code has been developed as part of the first year training programme for PhD students at the *EPSRC CDT in Sustainable Approaches to Biomedical Science: Responsible and Reproducible Research (SABS:R³) CDT* at the University of Oxford. The SABS:R³ programme focuses on providing comprehensive training in software development and software engineering to all of its students, in the context of industrially derived research in the biomedical sciences. To make this training immediately relevant, all students undertake an industry-supported group software development project over the course of their first year, allowing them to learn and apply their software skills in a realistic setting; the Epiabm software is the result of one of these year-long projects. The project was supported by colleagues in Roche's Infectious Disease Modelling Group.

Two simulation backends are provided: a pedagogical Python backend (with full functionality) and a high performance C++ backend for use with larger population simulations. Both are highly modular, with complete documentation and a range of working examples for ease of understanding and extensibility. The code is thoroughly tested, with 100% coverage on unit tests alongside a range of functional and integration tests. Epiabm is publicly available through GitHub at github. com/SABS-R3-Epidemiology/epiabm.

IMPLEMENTATION AND ARCHITECTURE

The basic units of an agent-based model are individual people. For each individual, personal characteristics such as age are used to determine their vulnerability to infection. Each individual is also assigned a time-dependent infection state, which is based on the basic SEIR (Susceptible/Exposed/Infected/Recovered) model but can be extended to include additional infection states and sub-states (e.g. hospitalised individuals). Individuals may move between states according to a network defined by a state transition matrix, depicted in Figure 1. The set of states and the network between them used within the CovidSim model is implemented by default in Epiabm, but specific methods to add user-defined compartments and connections allow for configurability to allow modelling of other diseases or alternative networks of states.

All individuals are initially 'Susceptible', a state where they have not been infected, and have no prior immunity

to the virus. Upon being infected, individuals initially become 'Exposed' (where they remain unable to infect others), before transitioning to an infected state after a randomly sampled latent time. The model implements a number of different 'Infected' states with differing degrees of severity, where vulnerable individuals have a higher likelihood of entering states with increased probabilities of death. This method also allows pressure on public health systems and intensive care units to be tracked, through specific states for those ill enough to require these facilities. Individuals with sufficiently severe infections may subsequently die, while all others will

enter the 'Recovered' status after a randomly sampled waiting time. While this model does not include waning immunity by default, the configurability of this method allows for simple addition of a 'Recovered' to 'Susceptible' pathway over sufficient time.

Each individual is a member of a population, with the overall population being made up of a hierarchy of cells and microcells as illustrated in Figure 2. Individuals are assigned to households and to places (such as schools or workplaces) within these microcells, through which they may infect other individuals. Such infection events are probabilistic and depend on a number of factors
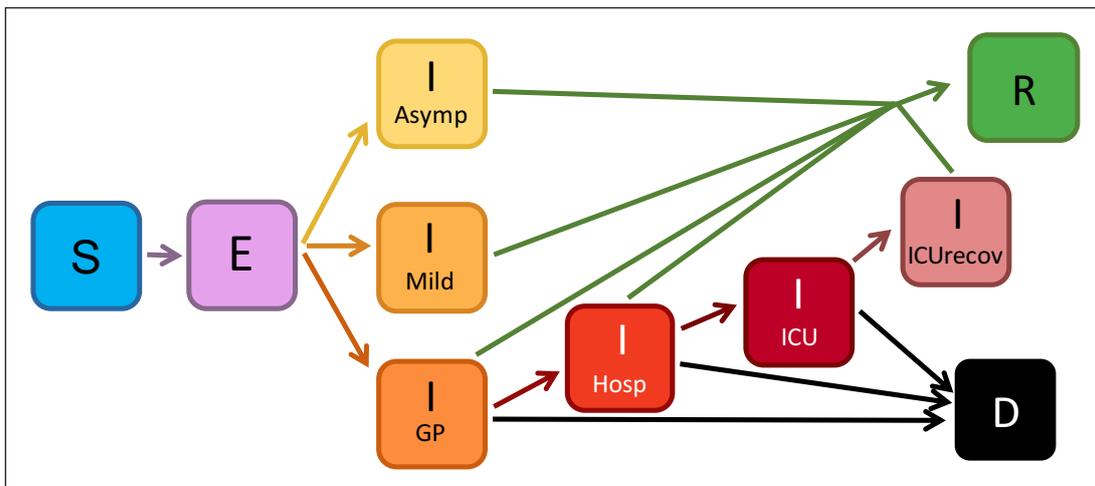

Figure 1 Infection state progression, with arrows depicting the possible routes of progression through this scheme. Based on a SEIRD model, where all individuals are either susceptible (S) to the disease, exposed (E) to the disease, infected (I) by the disease, or who have recovered (R) or died (D) from the disease. Multiple compartments for infection account for cases of differing severity, and this network is highly configurable, and new compartments or connections can be added with ease.
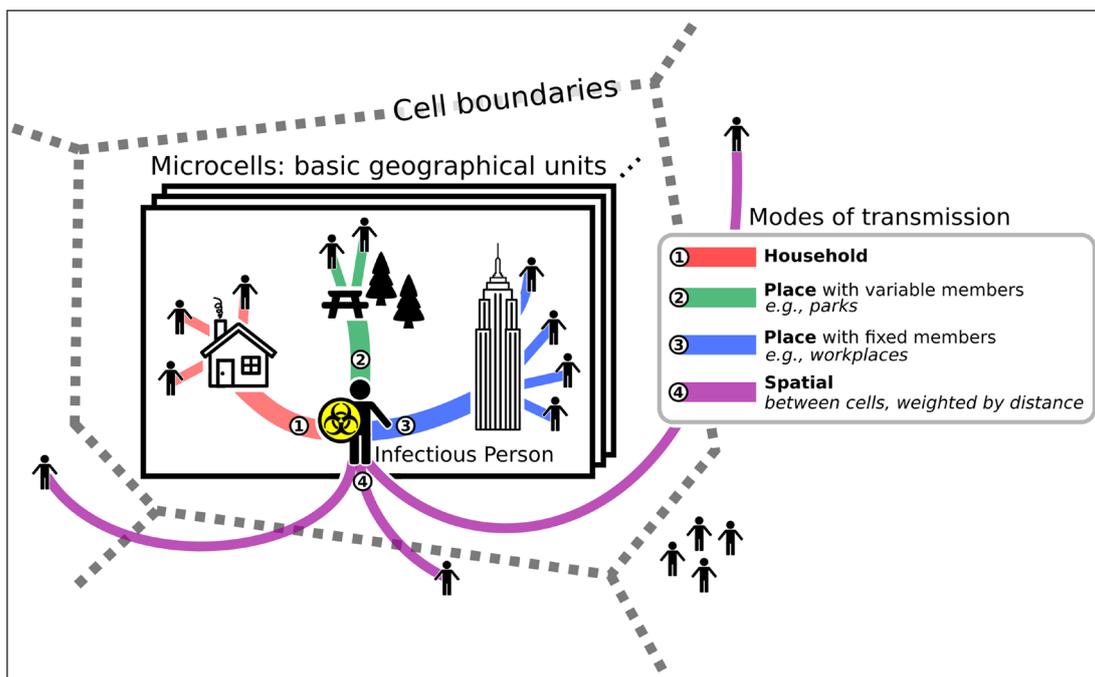

Figure 2 Population structure in Epiabm. A population is formed of many microcells, which are grouped into cells (the largest spatial unit). Infected individuals may infect others within their cell through households and places, while infections are spread between cells according to a spatial kernel.

including the age and infectiousness of the individuals involved.

Epiabm supports a complete epidemic simulation through a number of key workflow steps:

- Population Generation,
- Simulation Configuration,
- Simulation Evaluation,
- Results Output.

It also includes a range of example plotting scripts to visualise output data. Each of these workflow steps is explained below:

Population Generation
To generate a population, users can read in an external file with counts of places, households and individuals in each infection state per microcell (examples of this are provided in our repository). Alternatively, users may configure the population randomly based on given parameters (such as the number of places of each type per cell).

Simulation Configuration
A simulation may be configured by assigning a number of sweeps. These are functions that iterate over the population and are responsible for within-host infection progression and infection events between individuals via various transmission mechanisms (e.g. via households or places). A parameters file must also be specified here, with key-value pairs for each parameter used in the model – the default values provided are the same as those used in CovidSim, and referenced in our Wiki.

Simulation Evaluation
To evaluate the simulation, the sweep functions are called at each time step. This is completely modular, and so different infection mechanisms (such as via households or workplaces) may be removed independently to explore the role of different transmission mechanisms in the viral spread.

Results Output
At each timestep, output loggers may be used to record the current state of the simulation in a .csv file, over a range of resolutions – from microcell to global.

A full representation of the simulation routine is given in Appendix A.

COMPARISON TO COVIDSIM
While we have endeavoured to emulate both the overall structure and the functionality of CovidSim, the architecture we have used to achieve this in Epiabm differs significantly. Most notably, we have used a strongly object-oriented approach to population

generation and storage; while less efficient, we believe this is more intuitive and will enable other users to easily adapt sections of the code for their own use. While many aspects of simulation configuration (such as determining which infection sweeps to use) are specified through command line flags and parameter files in CovidSim, we have chosen to manage configuration though workflow scripts, again to increase readability and ease of sharing. This is made possible through our modular architecture; while CovidSim combines sweeps and output functions in large code blocks, Epiabm has separate individual classes and methods that may be included or excluded at will. Related classes, such as the different infection sweeps, also inherit from abstract parent classes to reduce code duplication where daughter classes have similar functionality.

EXAMPLE SIMULATION
An example simulation was configured using a synthetic population of 10,000 individuals distributed across 200 cells, each containing 2 microcells with five households per microcell. One infected individual is initialised in the central cell, with mild infection status. The resulting propagation of the infection through the population over a period of 80 days is visualised in Figure 3.

We also configured a national-scale simulation based on real-world parameter values to enable direct comparison to the results of CovidSim. The region of Gibraltar (with an approximate population of 34,000) was chosen for computational simplicity. Age-stratified output plots from both software packages are displayed in Figure 4 and show strong agreement, with the number of weekly cases peaking around April 15–April 22.

This simulation takes 42 seconds to run on the python backend, and 8 seconds on the C++ backend (including all population configuration and model running) on an AMD 3600X processor (6 cores, 3.8 GHz). In comparison, CovidSim takes a total of 45 seconds, although this is heavily dominated by build time (which does not scale as heavily in more complex cases) with the simulation alone running in 4.5 seconds. This highlights the performance compromises that have been necessary particularly in pyEpiabm, to ensure modularity and readability, relative to the highly optimised codebase of CovidSim. For this reason, the largest simulations we have conducted are for the country of New Zealand (with a population of 5 million people), which took 8 hours on the processor above. Despite this, we have demonstrated the capacity of Epiabm to perform complete epidemic simulations on small countries on a personal computer in feasible time periods, sufficient for many educational and research purposes.

Full code listings for these simulations are available on GitHub, alongside more basic workflows to introduce new users to the simulation capabilities of this software.
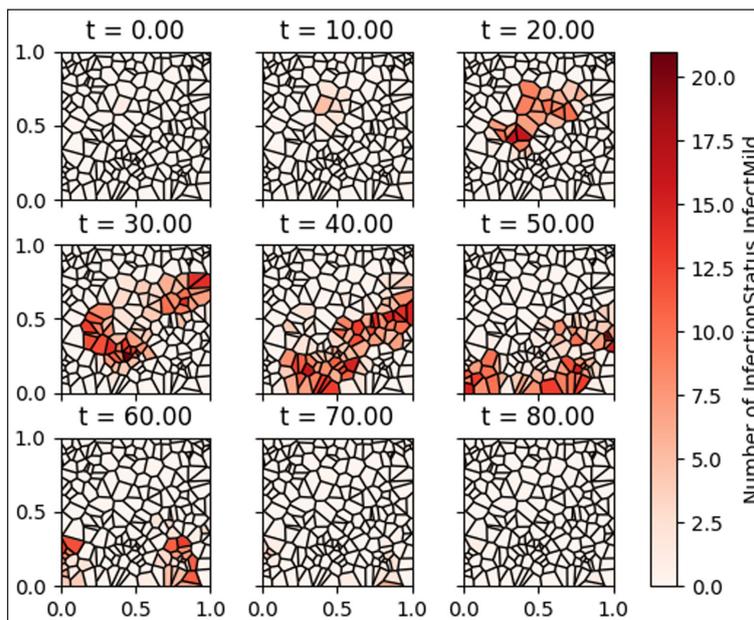
Figure 3 Spatial distribution of infected individuals within the population, at different time points during the simulation. Configured with a population of 10,000 people distributed across 200 cells, each containing 2 microcells with 5 households per microcell. One infected individual is initialised in the central cell, with mild infection status; the simulation is run for 80 days. Inter-cell infections only occur between nearby cells, allowing visualisation of the infection propagating through the simulation region over time.
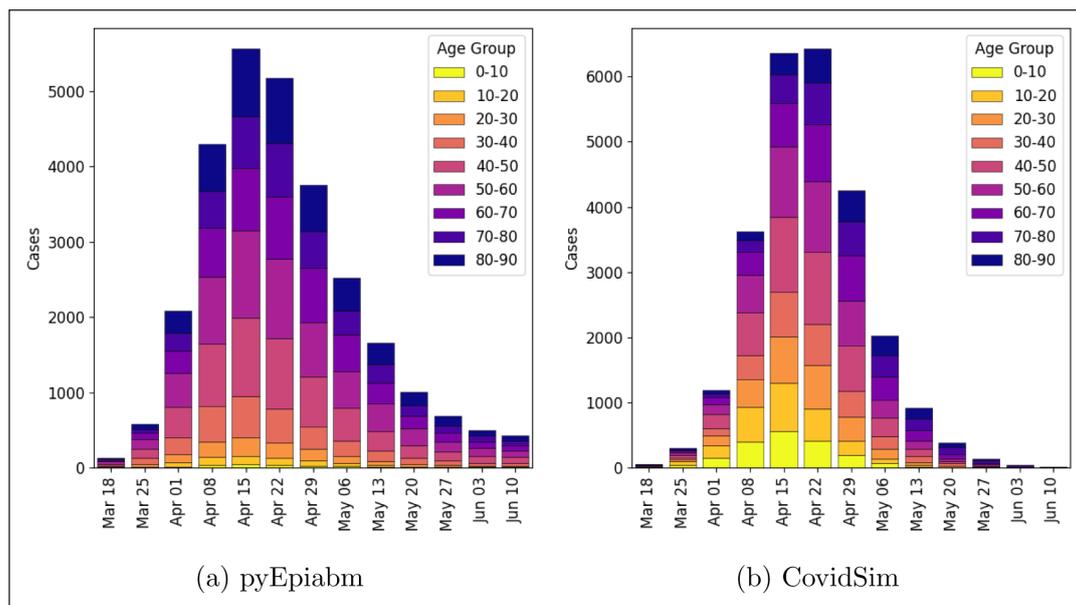

Figure 4 A comparison of simulation outputs from pyEpiabm **(a)** and CovidSim **(b)**, for an epidemic in Gibraltar initiated by 100 infected individuals. While the outputs are highly stochastic, a strong agreement is broadly observed.

QUALITY CONTROL

Both the Python and C++ backends have full unit testing with 100% test coverage, verifying expected behaviour of all deterministic and stochastic methods. Functional testing is used for automatic verification of random seed reproducibility in both population generation and simulation methods.

Epiabm also has testing routines to ensure all publicly exposed methods and classes are included in the documentation, and uses Flake8 linter tests to ensure that contributed code is consistent in style. All these tests are included in a continuous integration pipeline implemented through GitHub workflows, with unit testing evaluated across Python versions 3.6–3.9 and the latest macOS, Windows and Ubuntu distributions with Python version 3.8.

The CONTRIBUTING.md file in the epiabm repository contains more detailed and up-to-date information on our development workflow, testing and continuous integration infrastructure, and coding style guidelines.

Users are also provided with a number of example workflows, to configure and run different types of simulation, as well as plotting methods to visualise their outputs. This includes simple simulations with known

behaviour, to allow bench-marking against pre-existing models.

(2) AVAILABILITY

OPERATING SYSTEM
Epiabm uses no functions specific to any operating system (OS) and so can run on any OS that provides Python and C++.

PROGRAMMING LANGUAGE

- Python – version 3.6 or higher.
- C++ – version 17 or higher (for cEpiabm only, recommended compiler G++ 9)
- CMake – version 3.15 (for cEpiabm only)

ADDITIONAL SYSTEM REQUIREMENTS
Memory and disk space dependent on usage case.

DEPENDENCIES
Essential:

- numpy >=1.8
- packaging
- pandas >=1.4
- tqdm

Optional:

- flake8 >= 3 – Used to check code style
- matplotlib – Used in example workflow
- parameterized – Used in unit tests
- sphinx >= 1.5 – Used to generate documentation

LIST OF CONTRIBUTORS
Our project is hosted on GitHub, and publicly visible, allowing researchers from around the world to find our work and contribute to the codebase. In addition to the authors of this paper, who were the primary developers to Epiabm, we also acknowledge contributions from the following individuals:

- Open-Source Software Contribution – Saket Kumar, Netaji Subhash Engineering College, Maulana Abul Kalam Azad University of Technology, India.
- Open-Source Software Contribution – Pietro Monticone, University of Turin.

SOFTWARE LOCATION
Archive
   Name: Zenodo
   Persistent identifier: DOI: 10.5281/zenodo.7327444
   Licence: BSD 3-Clause

Publisher: Kit Gallagher
Version published: 1.0.1
Date published: 16/11/22

Code repository
   Name: GitHub
   Persistent identifier: https://github.com/SABS-R3-Epidemiology/epiabm
   Licence: BSD-3-Clause
   Date published: 01/03/22

LANGUAGE
English

(3) REUSE POTENTIAL

Epiabm is designed with both research and educational use in mind. The modular aspect of the code allows for highly configurable simulations and investigation into the sensitivity of large scale agent-based models to different transmission mechanisms. While the default parameter values provided are tailored to the spread of COVID-19 within the UK population, minimal reconfiguration is required to adapt these for other countries/diseases. The modular nature of the code offers considerable freedom to explore and compare the roles of different transmission mechanisms in epidemic growth for viral strains with varied properties. The model can also be extended to include custom interventions, both pharmaceutical and non-pharmaceutical, on local and global spatial scales, and indeed this will be the task of one the first year group projects in the coming academic year.

Detailed documentation and example workflows are provided, also enabling use of Epiabm in educational settings and for users with little familiarity with agent-based epidemiological models. We welcome questions, suggestions, bug reports, and user contributions via the GitHub repository, which acts as a central communication platform for Epiabm. A detailed guide on contributing to Epiabm is also available there.

APPENDIX A
EPIABM IMPLEMENTATION
This section introduces our implementation of CovidSim, in the pyEpiabm backend. The cEpiabm backend broadly follows the same structure at all levels, and all pseudocode given here also applies to this backend.

The overall workflow is illustrated in Algorithm 1, which represents the simulation flow file and the broad functionality of the '`Simulation`' class. The bulk of the simulation work occurs within the '`sweep()`' function calls, an example of which is given in Algorithm 2.

Require: parameters.json file
Ensure: Output .csv files   ▷ Containing case numbers at each timestep
   pop_params ← {'population_size': 1000, ...}
   population ← ToyPopulationFactory.make_pop(pop_params)

   sim ← Simulation()
   sim.configure(population, list_of_sweeps)
   t ← start_time
   while t < end_time **do**
      HouseholdSweep()
      PlaceSweep()
      t ← t + time_step
   end while

Algorithm 1 Outline of the overall workflow for Epiabm.

Require: population, t (current time)
Ensure: Queues individuals newly infected within households
   for cell in population.cells **do**
      for infector in cell.persons **do**
         assert infector.household is not None:
         if not infector.is_infectious() **then**
            continue
         end if

         for infectee in infector.household.persons **do**
            if not infectee.is_susceptible() **then**
               continue
            end if
            force_of_infection ← household_foi(infector, infectee, t)
            r ← random.uniform(0, 1)
            if r < force_of_infection **then**
               cell.enqueue_person(infectee)   ▷ Queue person for infection
            end if
         end for
      end for
   end for

Algorithm 2 Example of an infection sweep from Epiabm, handling infections occurring within a household.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR CONTRIBUTIONS

Kit Gallagher, Ioana Bouros, Nicholas Fan, Elizabeth Hayman, Luke Heirene and Patricia Lamirande contributed equally to this research.

AUTHOR AFFILIATIONS

Kit Gallagher orcid.org/0000-0003-1401-115X
Mathematical Institute, University of Oxford, UK

Ioana Bouros ⓘ orcid.org/0000-0002-5532-3603
Department of Computer Science, University of Oxford, UK

Nicholas Fan
Mathematical Institute, University of Oxford, UK

Elizabeth Hayman ⓘ orcid.org/0000-0002-3689-1242
Department of Engineering Science, University of Oxford, UK

Luke Heirene ⓘ orcid.org/0009-0000-2845-7860
Mathematical Institute, University of Oxford, UK

Patricia Lamirande ⓘ orcid.org/0000-0001-6585-4734
Mathematical Institute, University of Oxford, UK

Annabelle Lemenuel-Diot ⓘ orcid.org/0000-0002-9837-8188
Roche Pharmaceutical Research and Early Development,
Pharmaceutical Sciences, Roche Innovation Center Basel,
Switzerland

Ben Lambert ⓘ orcid.org/0000-0003-4274-4158
College of Engineering, Mathematics and Physical Sciences,
University of Exeter, UK

David Gavaghan ⓘ orcid.org/0000-0001-8311-3200
Department of Computer Science, University of Oxford, UK

Richard Creswell ⓘ orcid.org/0000-0002-9491-1897
Department of Computer Science, University of Oxford, UK

REFERENCES

1. **Adam D.** Special report: The simulations driving the world's response to COVID-19. *Nature*. Apr. 2020; 580(7803): 316–318. DOI: https://doi.org/10.1038/d41586-020-01003-6

2. **Birrell P,** et al. Real-time nowcasting and forecasting of COVID-19 dynamics in England: the first wave. *Philosophical Transactions of the Royal Society B: Biological Sciences*. 2021; 376(1829). DOI: https://doi.org/10.1098/rstb.2020.0279

3. **Crick T, Hall BA, Ishtiaq S.** Reproducibility in Research: Systems, Infrastructure, Culture. *Journal of Open Research Software*. Nov. 2017; 5. DOI: https://doi.org/10.5334/jors.73

4. **Danon L,** et al. A spatial model of COVID-19 transmission in England and Wales: early spread, peak timing and the impact of seasonality. *Philosophical Transactions of the Royal Society B: Biological Sciences*. May 2021; 376(1829). DOI: https://doi.org/10.1098/rstb.2020.0272

5. **Davies NG,** et al. Effects of non-pharmaceutical interventions on COVID-19 cases, deaths, and demand for hospital services in the UK: a modelling study. *The Lancet Public Health*. 2020; 5(7): e375–e385. ISSN: 2468–2667. DOI: https://doi.org/10.1016/S2468-2667(20)30133-X

6. **Eglen SJ.** CODECHECK certificate 2020-010. en (2020). DOI: https://doi.org/10.5281/ZENODO.3865490

7. **Ferguson NM,** et al. Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand; 2020. DOI: https://doi.org/10.25561/77482

8. **Ferguson NM, Nedjati Gilani G, Laydon, D.** *COVID-19 CovidSim microsimulation model*. Computer Software. R Foundation for Statistical Computing; Apr. 2020.

9. **Ferguson NM,** et al. Strategies for mitigating an influenza pandemic. *Nature*. Apr. 2006; 442(7101): 448–452. DOI: https://doi.org/10.1038/nature04795

10. **He S, Peng Y, Sun K.** SEIR modeling of the COVID-19 and its dynamics. *Nonlinear Dynamics*. June 2020; 101(3): 1667–1680. DOI: https://doi.org/10.1007/s11071-020-05743-y

11. **Hunter E, Mac Mamee B, Kelleher JD.** A Comparison of Agent-Based Models and Equation Based Models for Infectious Disease Epidemiology; 2018. DOI: https://doi.org/10.18564/jasss.3414

12. **Keeling MJ,** et al. Predictions of COVID-19 dynamics in the UK: Shortterm forecasting and analysis of potential exit strategies. *PLOS Computational Biology*. Jan. 2021; 17(1): 1–20. DOI: https://doi.org/10.1371/journal.pcbi.1008619

13. **Lyra W,** et al. COVID-19 pandemics modeling with modified determinist SEIR, social distancing, and age stratification. The effect of vertical confinement and release in Brazil. *PLOS ONE*. Sept. 2020; 15(9): e0237627, Samy AM (ed.). DOI: https://doi.org/10.1371/journal.pone.0237627

14. **Shen C, Taleb NN, Bar-Yam Y.** Review of Ferguson et al 'Impact of non-pharmaceutical interventions...'. *New England Complex Systems Institute*; Mar. 2020.

15. **Stodden V, Miguez S.** Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research. *Journal of Open Research Software*. July 2014; 2(1). DOI: https://doi.org/10.5334/jors.ay

16. **Wooldridge M.** *The Software that Led to the Lockdown*. July 2020.

17. **Wu JT, Cowling BJ.** The use of mathematical models to inform influenza pandemic preparedness and response. *Experimental Biology and Medicine*. Aug. 2011; 236(8): 955–961. DOI: https://doi.org/10.1258/ebm.2010.010271

]u[

]u[