BharatSim: An agent-based modelling framework for India

Philip Cherian[1], Jayanta Kshirsagar[2], Bhavesh Neekhra[3], Gaurav Deshkar[2], Harshal Hayatnagarkar[2], Kshitij Kapoor[3], Chandrakant Kaski[2], Ganesh Kathar[2], Swapnil Khandekar[2], Saurabh Mookherjee[2], Praveen Ninawe[2], Riz Fernando Noronha[1], Pranjal Ranka[2], Vaibhhav Sinha[1,3], Tina Vinod[2], Chhaya Yadav[2], Debayan Gupta[4], Gautam I. Menon[1,5,6,7]

1 Department of Physics, Ashoka University, Plot No. 2, Rajiv Gandhi Education City, NCR P.O.Rai, Sonepat, Haryana 131 029 INDIA
2 Engineering for Research (e4r), Thoughtworks Technologies, 6th Floor, Binarius, Deepak Complex, Shastri Nagar, Yerwada, Pune, Maharashtra 411 006, INDIA
3 Simons Centre for the Study of Living Machines, National Centre for Biological Sciences, Tata Institute of Fundamental Research, Bangalore, Karnataka, India
4 Department of Computer Science, Ashoka University, Plot No. 2, Rajiv Gandhi Education City, NCR P.O.Rai, Sonepat, Haryana 131 029 INDIA
5 Department of Biology, Ashoka University, Plot No. 2, Rajiv Gandhi Education City, NCR P.O.Rai, Sonepat, Haryana 131 029 INDIA
6 The Institute of Mathematical Sciences, CIT Campus, Taramani, Chennai 600 113 INDIA
7 Homi Bhabha National Institute, BARC Training School Complex, Anushaktinagar, Mumbai, Maharashtra, India 400 094 INDIA

Abstract

BharatSim is an open-source agent-based modelling framework for the Indian population. It can simulate populations at multiple scales, from small communities to states. BharatSim uses a synthetic population created by applying statistical methods and machine learning algorithms to survey data from multiple sources, including the Census of India, the India Human Development Survey, the National Sample Survey, and the Gridded Population of the World. This synthetic population defines individual agents with multiple attributes, among them age, gender, home and work locations, pre-existing health conditions, and socio-economic and employment status. BharatSim's domain-specific language provides a framework for the simulation of diverse models. Its computational core, coded in *Scala*, supports simulations of a large number of individual agents, up to 50 million. Here, we describe the design and implementation of BharatSim, using it to address three questions motivated by the COVID-19 pandemic in India: (i) When can schools be safely reopened given specified levels of hybrid immunity?, (ii) How do new variants alter disease dynamics in the background of prior infections and vaccinations? and (iii) How can the effects of varied non-pharmaceutical interventions (NPIs) be quantified for a model Indian city? Through its India-specific synthetic population, BharatSim allows disease modellers to address questions unique to this country. It should also find use in the computational social sciences, potentially providing new insights into emergent patterns in social behaviour.

Author summary

Agent-based simulations provide granular ways of describing the dynamics of each individual in a population. Such models are especially useful in describing the spread of an infectious disease, since they can be used to incorporate individual-level heterogeneity in behaviour and susceptibilities, as well as spatio-temporal information. BharatSim is such an agent-based modelling framework for India. It can simulate populations at multiple scales, from a few hundreds to several millions. It creates and uses a predefined synthetic population for India, assimilating it into a simulation framework. The synthetic population defines individuals with multiple attributes, among them age, sex, home, and work locations. We demonstrate the use of BharatSim in three contexts related to the COVID-19 pandemic in India.

1 Introduction

Mathematical and computational approaches to the dynamics of an infectious disease have a long history. Early quantitative approaches to this problem were associated with the names of Bernoulli, Hamer, and Ross [1–3]. Around a hundred years ago, Kermack and McKendrick [4] used a set of coupled non-linear ordinary differential equations to describe how the numbers of susceptible (S), infectious (I) and recovered (R) individuals in a population changed over time. Their model for the dynamics of an infectious disease is referred to as the SIR model. In this model, individuals are assigned to "compartments" labelled S, I and R [5]. Variants of the SIR model that incorporate additional compartments, such as Exposed (E) or Vaccinated (V), have been customized to different infectious diseases as well as to varied interventions [6].

Compartmental models are valuable tools for the description of infectious disease dynamics. They can be formulated in terms of coupled non-linear ordinary differential equations (ODEs) and solved numerically with relative ease. However, in their most-used formulation, they are mean-field (also called "well-mixed") models, although systematic improvements on this approximation are possible. By construction, mean-field models are deterministic, and exclude information contained in real-world networks of physical interactions and in spatial variation [6]. Social factors, such as family structures, socioeconomic status, and contact networks, are difficult to incorporate into such models and they are particularly ill-equipped to describe how population-level shifts emerge from subtle modifications in individual behaviour. The inability to easily describe the many independent sources of stochasticity that are relevant to disease dynamics is another limitation of these models.

Achieving a more comprehensive understanding of disease spread and its social determinants requires model approaches that can go beyond the assumption of a well-mixed population [7–12]. At the most granular level, one could describe each separate individual in a population. Agent- and network-based models implement such granularity, with individuals in these models capable of modulating their behaviour in response to how they perceive their environment [13–16], including information from their contact networks [17–19]. Such models can also incorporate geo-spatial data [20].

This increased flexibility comes at a cost. Both network- and agent-based models must use many more assumptions than compartmental models require. The parameters that enter these models are often imperfectly known and model output may be sensitive to the values chosen. Studying such models also requires significant amounts of numerical computation. On the positive side, however, more realistic investigations of disease dynamics in specific populations become possible [21]. Targeted interventions, such as lockdowns, school closures, vaccination drives, and restrictions on public transport, can be assessed more realistically than is possible with ODE-based

approaches [22, 23]. An exponential expansion in computational power over the past several decades has also made simulating large-scale individual-level models progressively easier.

A number of agent-based models have been used to study disease dynamics [24–28]. Relatively few such models, though, have been customized for India [29, 30]. One, the IISc-TIFR city-scale agent-based simulator [31], has been used to model COVID-19 spread in the major Indian cities of Bengaluru and Mumbai. This simulator has been used to evaluate strategies for the reopening of public transport in the background of an epidemic. It has also been used to study lockdowns and related interventions. Another example of a Indian city-scale simulator is *EpiRust* [32, 33], which has been used to simulate the cities of Mumbai and Pune.

Here, we describe BharatSim, an open-source, ultra-large-scale, agent-based framework for the simulation of infectious diseases. It is accompanied by a synthetic population that is customised to India, although it is easily generalizable to other countries and contexts. BharatSim allows users to easily adapt their models to different infectious diseases without having to modify a complex preexisting code base. Memory requirements and computational time in agent-based models increase non-linearly with the number of agents and the detail with which they are described – one must run such simulations multiple times to obtain statistically significant results. One should be able to efficiently scale up a model without compromising on performance. The BharatSim simulation engine is designed keeping such requirements in mind. Finally, BharatSim's formulation as a simulation modeling framework, as opposed to an application, insulates the user from the details of its implementation, while providing them flexibility in the construction of models.

A typical user of BharatSim will specify a model, initialize it with input data, run the simulation, and then analyze its results, often visually. BharatSim's design incorporates three components: a synthetic population designed for India, a simulation engine, and a visualization engine. A user might use the simulation engine to construct and simulate a model of disease spread on any given synthetic population. They could then visualize the output from the simulation using the visualization engine. However, these components can also be used independently.

In this paper, we describe the generation of the synthetic population, and the design of the simulation and visualization engines that constitute BharatSim. To validate our results, we compare statistical measures applied to the synthetic population to aggregated data obtained from official sources. We describe the different components of the simulation engine and show how it can be used by a typical user to create a customized model. We detail the features of the visualization engine, including the variety of spatio-temporal data that can be imported and visualized. We then explore how policy measures for COVID-19 can be assessed by simulating multiple scenarios, including counter-factual ones.

We apply BharatSim to three questions: (i) Under what conditions can schools be safely reopened given specified levels of hybrid immunity?, (ii) How do new SARS-CoV-2 variants that evade prior immunity alter disease dynamics? and (iii) How do we quantify the effects of varied NPIs on disease spread in a model Indian city? For this, we use disease states corresponding to a specific compartmental model for COVID-19, INDSCI-SIM [34]. This model has been benchmarked in considerable detail using data for the first wave of COVID-19 in India [34].

We show that at reported levels of seropositivity in India, and given the relative mildness of COVID-19 infections in the younger population, schools across India could have been reopened as early as August of 2021 with only a marginal increase in the number of cases. Further, our models show that the relative mildness of the Omicron wave was a consequence of hybrid immunity. Finally, we describe the impacts of

targeted NPIs such as lockdowns and vaccination drives. We show that, when judiciously applied, they contribute significantly to slowing the progress of a disease.

2 Design and implementation

BharatSim is designed to be usable on a range of hardware, from personal laptops to High Performance Computing (HPC) clusters. Given that one application of BharatSim would be the simulation of an average Indian state ($\sim$ 50 million agents), it is designed to handle large population sizes without significant overhead or degradation in speed (see Appendix S1). Its design allows modellers from a range of backgrounds, even those lacking significant programming experience, to be able to easily run existing models, tweak them and extend them in new ways. BharatSim is accessible at the site: bharatsim.ashoka.edu.in.

2.1 Synthetic population

BharatSim's synthetic population models those details of the actual population that are relevant to the interactions between agents. We use a variety of data sources to generate a population of individuals and households with demographic attributes that are statistically similar to survey data, using statistical methods and machine learning algorithms. These can be used to generate data at various scales, ranging from small communities to entire states [35].

2.1.1 Datasets and data preparation

The primary sources of data for our algorithms include the Census of India, the India Human Development Survey (IHDS), the National Sample Survey (NSS), and the Gridded Population of the World (GPW). We use sample survey data (microdata) from India Human Development Survey-II [36], marginals from Census, 2011 [37], data for employment from NSS [38], and population density from GPW grid population density dataset [39]. A hybrid process is used to scale up these limited datasets to the size of a population. The data is initially curated to remove obvious inconsistencies. Then, the customized hybrid of statistical and machine learning techniques detailed below is used to generate new data on the required scale. Table 1 lists the different datasets and the role they play in the generation of the synthetic population.

The IHDS-II (2011-12) [36] is a nationally representative, multi-topic survey of 42,152 households in 1,503 villages and 971 urban neighbourhoods across India. We use microdata from two datasets from this survey, *Individual* and *Household*, in the machine-learning model described below. The Census of India 2011 was performed across 28 states and 8 union territories, covering 640 districts, 5,924 sub-districts, 7,935 towns and more than 600,000 villages. We use marginal data from Census 2011 as input to the Iterative Proportional Updating (IPU) method described below, since the Census microdata is not publicly available. The NSS collects data through nationwide sample household surveys on various socio-economic subjects. We use employment data from this survey. GPW provides spatial population maps for all countries and their sub-divisions – we use population density data for India to place the generated synthetic population in a geographical context. The next section describes the use of these different datasets in our hybrid model to generate a synthetic population for a district of India.

Source	Description	Use in model
IHDS [36]	State-wise microdata	Generating attributes for individuals and families
Census [37]	District-wise marginal data	Generating attributes for individuals and families
NSS [38]	District-wise data for employment	Estimating homebound population
GPW [39]	Population density data	Assigning geo-locations to households and other locations

Table 1. Source datasets for generating the synthetic population. The different data sources used are specified, along with the role they play in the generation of the synthetic population.

2.1.2 Population generation

Our objective is to generate an accurate synthetic population for India with joint distributions of relevant variables that statistically match those of the actual population. The synthetic population should capture relevant attributes of individuals in the real population, such as the distributions of age, height, weight, and sex. The synthetic population must also incorporate social network structures appropriate to households and workplaces. We might also require the distribution of family sizes, the joint distribution of age and sex of individuals within a family, the geographical distribution of households, workplaces, and schools within administrative units (cities, districts, and states), and the number of people who are associated with a workplace or school. The population generation pipeline described below is illustrated in Fig 1.

Iterative Proportional Updating (IPU) [40] is a statistical sampling technique that accepts both information regarding marginals from the real population, and individual-level data from a sample of the real population. IPU is used to iteratively scale up the sample to generate a large-scale population, ensuring that the marginal distributions for both households and individuals in the synthetic population match with the known marginal distributions in the real population.

While IPU generates a large-scale population in which households have a realistic distribution of the number of family members and their age and sex, the joint distribution of various attributes in the synthetic population should also resemble those of the real population. To achieve this, we use a machine learning technique, Conditional Tabular Generative Adversarial Network (CTGAN) [41], to add individual-level attributes for which marginals are not available. CTGAN is trained using data from a sample survey and used to generate highly correlated individual attributes like heights, weights, and comorbities. This provides a reasonable joint distribution of various individual-level attributes. Additionally, different trained instances of our CTGAN model can also be used to combine different survey datasets with an overlapping set of attributes.

A realistic geographical distribution of the population must describe the spatial distribution of households, workplaces, schools, and public places, and the number of individuals that frequent them. We use grid population density data to assign individuals to specific locations. A row in the grid population density data consists of latitude (X), longitude (Y) and the number of individuals (Z) who live within a square of side 1 km, centered at (X, Y). We use reject-sampling to assign geo-location points within a geographical boundary, faithful to population density distributions [39]. An individual is then assigned locations such as workplaces, schools and public places, based on a probability inversely proportional to their distances from the individual's household [42].

The sampling method determines the geographical boundaries of the region for which data is being generated using a polygon provided by a geojson file. We filter the population density dataset by only retaining those rows for which the latitude X and

longitude Y are within the boundary polygon of the region. We sample, with replacement, a large number of latitude-longitude pairs from this filtered subset, using the number of individuals (Z) as the weight for this sampling. For a given row in the sample, we then add uniformly distributed random noise to both the latitude X and longitude Y. We reject those latitude-longitude pairs that are not within the polygon, until the required number of points is reached. Given that we do not have access to workplace distribution data, we make the natural assumption that it follows the population density distribution as well. Our technique can use actual workplace density data to obtain distributions more consistent with reality.

As an example, Fig 2 visualizes the geographic distribution of synthetic households, workplaces and schools in the combined districts of Mumbai and Mumbai Suburban.

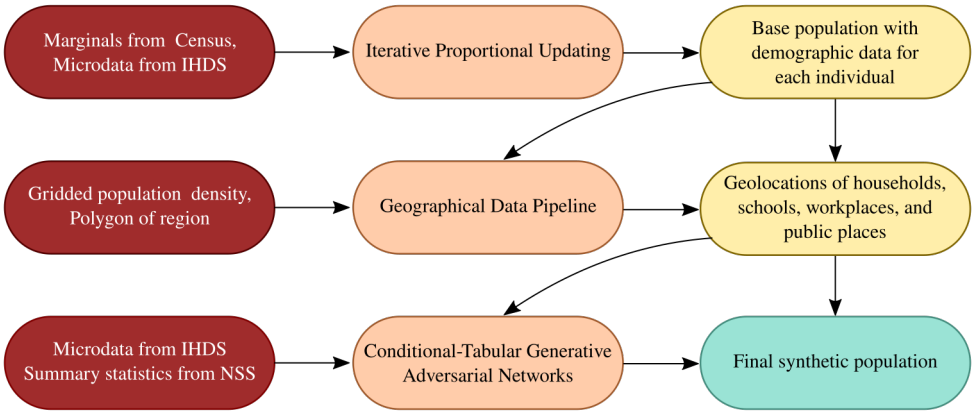

Fig 1. Pipeline for synthetic population generation. Census, gridded population, and survey data are first processed and subsequently used as the input to generate the final synthetic population.

Individuals in our synthetic population are associated with external locations which include workplaces, schools, and public places. These locations define their contact networks. All individuals are assigned job descriptions. This job description is sampled with replacement from the list provided in the IHDS-II dataset. Individuals below the age of 3 are "Homebound". For individuals above the age of 3 and below the age of 18, "Student" is assigned as the job description. The number of workplaces, schools and public places varies for each district. Each workplace has three attributes: workplace type, latitude and longitude. Just as for the job description, the attribute "workplace type" is generated by sampling (with replacement) from the list of jobs in the IHDS-II subset. Specific workplaces are assigned as schools by looking at a subset of workplaces of individuals with job descriptions set to "Teacher". Students are assigned to schools and adults to workplaces, provided they are not homebound. We assume that agents are more likely to visit public places which are closer to their homes.

Given an individual with home latitude X_h and longitude Y_h and a list of k possible external locations $\{E_k\}$ with latitude and longitude pairs $(X_{E_1}, Y_{E_1}), \ldots, (X_{E_k}, Y_{E_k})$, we calculate the Euclidean distance r between the home of the individual and the external location. Among those who visit external locations, a fixed fraction is labelled as taking public transport. (Although we do not explore this in this paper, such a label would allow us to examine the possibility of increased transmission in such crowded locations. This has been examined in prior work [31].) We then choose a real-valued function $f(r)$ which is strictly decreasing and non-negative in the positive real domain, and weight the probability of an individual being assigned the external location by the value $f(r)$. For our analysis, we choose $f(r) \propto 1/r^2$ so that the probability of an

external location being assigned is inversely proportional to the square of Euclidean distance between the individual's house and the external location. A geodesic distance metric can also be used, but the difference is insignificant at these scales.

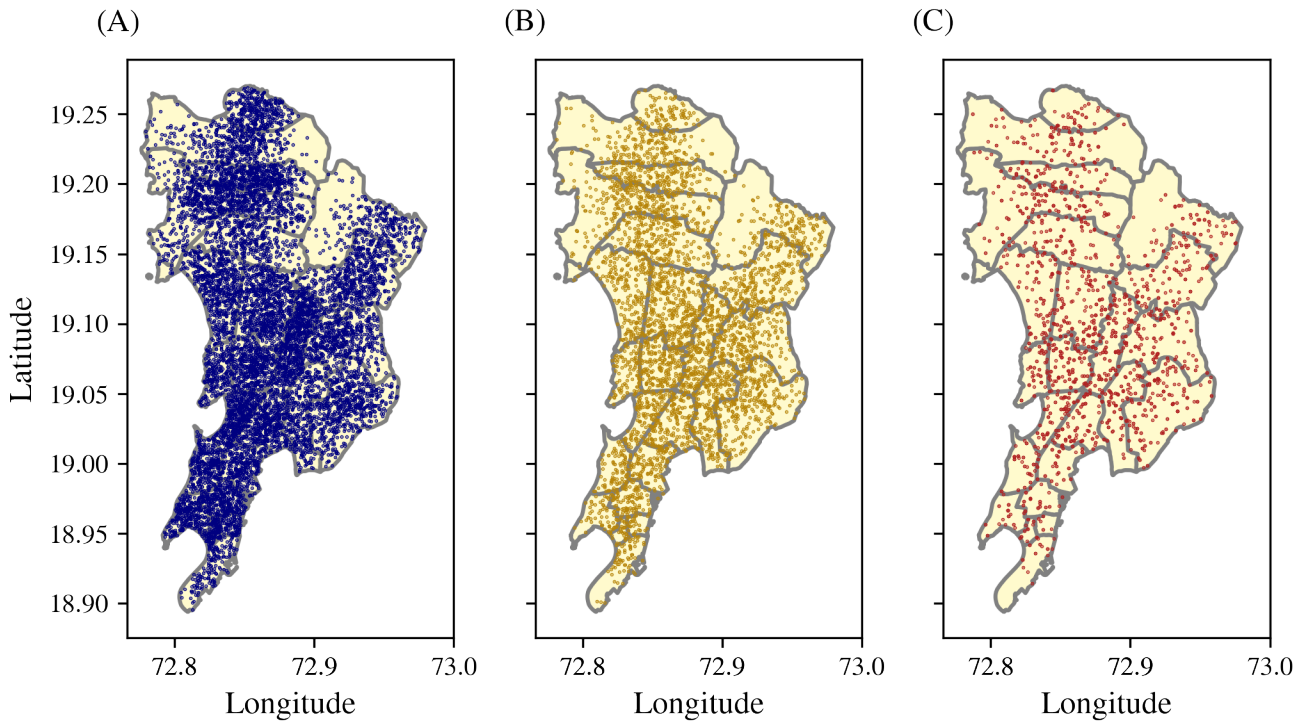

Fig 2. Distribution of geo-locations. Geographical distribution of (A) households, (B) workplaces, and (C) schools for the combined synthetic population of the districts of Mumbai and Mumbai Suburban.

2.1.3 A synthetic population for Mumbai

Located in the state of Maharashtra in India, Mumbai is one of India's largest cities, with a population of more than 12 million. We use a subset of the IHDS-II dataset obtained by filtering for individuals and households which are situated in the state of Maharashtra. (We use the filtered datasets for Maharashtra because the Mumbai dataset in IHDS-II contains a few hundred samples, which are not enough to be able to generate a quality synthetic population.) Each job description is drawn from the empirical distribution observed in the IHDS-II subset for Maharashtra.

Next, we generate synthetic workplaces, schools, and public places for our agents as detailed in previous sections. The final synthetic population has demographic data (age, height, etc.), disease data (chronic heart disease, diabetes, etc.), family data, geographic location for workplaces, schools etc., and socio-economic data (job label, religion, etc.). In total, we generate a synthetic population with 44 attributes.

2.1.4 Population benchmarks

We evaluate the synthetic population based on the following three criteria: (i) does the distribution of individual features in synthetic population match those in sample

surveys? (ii) do the joint distributions of selected features in synthetic population match those in sample surveys? and, (iii) can our synthetic population replace survey data in a machine learning regression task?

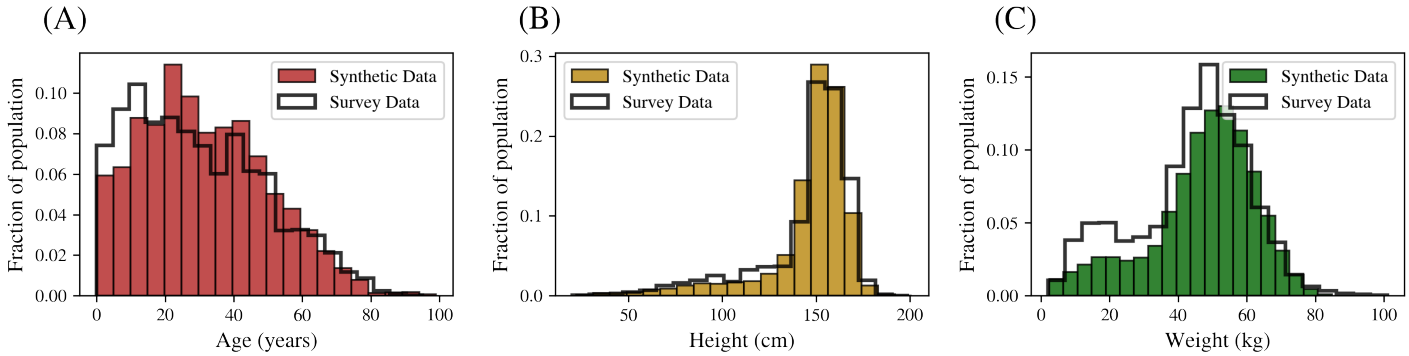

Fig 3. Histograms of marginal distributions. The distributions of age, height and weight in the synthetic population for the combined districts of Mumbai and Mumbai Suburban. In each case, this distribution is plotted over the distributions obtained from survey data, and these distributions compared using the Kolmogorov-Smirnov test. The results and KS statistic for each case are given in Appendix S2.

Fig 3 compares the marginal distributions of attributes of age, height, and weight across samples of the real and the synthetic population for the combined districts of Mumbai and Mumbai Suburban. We work with a randomly chosen subset comprising 10,000 individuals to compare with the survey data, although our synthetic population has 12 million individuals.

To benchmark the population, we compared the marginals of individual features (age, height, weight, sex) in the synthetic population with those in the survey data to quantify the similarities of their underlying statistical distributions. The results of our tests are shown in Appendix S2.

To examine whether the synthetic data could reproduce correlations present in the survey data, we use a machine-learning regression task. We first attempt to predict the weight based on age, sex and height. Similarly, we attempt to predict the height based on age, sex, and weight. We use two machine-learning models: Linear Regression and Multi-layer Perceptron (MLP) Regression [43–45]. The results of these regression tasks are reported in Table 2. We begin by training each model on a subset of the survey data. We choose this subset to be 80% of the entire sample. We then report the accuracy of the model on the remaining 20% of the sample. Next, we instead train our models with the entire synthetic population, and again report the accuracy of the model on the 20% of the survey data. The results of these comparisons are shown in Table 2.

2.2 The simulation engine

The simulation engine framework allows modellers to directly specify their models using a domain-specific, high-level language. This domain-specific language (DSL) is based on the *Scala* programming language [46], allowing modellers to extend their knowledge of *Scala* when creating their models. The DSL provides abstractions and constructs that make it easier for modellers to define general agent-based models of arbitrary complexity. It simplifies the creation of customized data structures, the execution of scenarios and experiments, as well as the integration with a synthetic population. Modellers can thus focus on the specific problem they wish to address even without being highly computationally skilled.

Machine-learning Test	Source Population	Synthetic Population
Linear Regression for Weight	0.72	0.69
MLP Regressor for Weight	0.78	0.76
Linear Regression for Height	0.69	0.65
MLP Regressor for Height	0.71	0.72

Table 2. Benchmarking synthetic population quality. To evaluate if the synthetic population can replace the real population, we compare different prediction models. In each model, we attempt to predict the weight (height) based on age, sex and height (weight). First, the model is trained and tested on the real population. Then, the same model is trained on synthetic population and tested on real population. Comparative results for Linear Regression and MLP Regression are shown.

Agents The `Agent` class encapsulates an individual agent's attributes, behaviours, and relations with other agents or entities. Modellers can extend the framework's inbuilt `Agent` class as required, including behaviors specific to the context of the problem they are addressing. For example, the `StatefulAgent` described below is one such inbuilt extension of the `Agent` class.

Behaviour An agent's behaviour is defined as an action that is executed at every simulation timestep (or "tick"). For example, an agent might have some probability of getting themselves vaccinated at each time-step. This probability can depend on their attributes like their age, socio-economic status, and prior vaccination history.

Stateful Agents A `StatefulAgent` is a specialized extension of the `Agent` class that supports the framework's Finite State Machine (FSM). This is particularly useful in models where agents can exist at any given time in only one "state", but where transitions exist between these states. The `StatefulAgent` interface can define and control these transitions between states. These include actions that must be performed when agents enter or exit these states, as well as actions that are performed in every simulation tick.

Networks BharatSim is a network-based simulation model, where networks determine agent interactions. Every `Agent` or a `StatefulAgent` is a `Node` and a `Network` is composed of such `Node`s, with `Relation`s between them. The framework allows the modeller to define relations between agents that mimic real life network locations like houses, offices, restaurants, or other public places. These networks can, in principle, be dynamic.

Relations Different `Nodes` can be related to each other using framework-defined as well as extendable `Relation`s. For example, an agent can be linked to his or her house, as well as his or her workplace. Two agents or workplaces can also be related to each other in this fashion.

Schedules Schedules define the `Network` locations of `Agent`s at a given instant in the simulation. For example, the `Schedule` can specify that a specific `Agent` is part of her "House" network for 0 to 8 simulation ticks and "Office" network for 9 to 18 ticks. The BharatSim DSL allows for the definition of a custom `Schedule` using predefined units (`Day`, `Week`, `Month`), but also allows modellers to define custom units of time, as per their requirement.

Interventions Interventions are a set of externally defined rules imposed on the agents, such as lockdowns or vaccination drives. BharatSim allows for the incorporation

of such interventions in order to study a range of counterfactual scenarios. Modellers can define multiple types of interventions, with different activation, deactivation, and reactivation conditions over the course of the simulation.

Simulation In BharatSim, model developers have to define a `Simulation` in order to be able to specify input data, register agents, define schedules, and generate output. Along with monitoring clock ticks, the `Simulation` stores all common data in a `Context`, which is shared globally and from which other objects like agents can query information.

2.2.1 Creating Agents and Networks

The simulation engine accepts the synthetic population as a CSV file. This data is stored in a *graph* data structure, a network of nodes related to each other through user-defined relations. These nodes represent both individual agents as well as network locations such as households or offices. The framework defines a `Node` class which allows for relations to be established between other nodes. Using such a data structure makes the framework flexible and domain-independent. This can be implemented in one of two ways, either by using Neo4j, a graph database, or using the *Scala* programming language's scalable map implementation, `TrieMap`. Both these structures optimize data operations, allowing the simulation to scale efficiently to larger populations.

Scala allows programmers to define their own datatypes, justifying its choice for the BharatSim simulation framework. In particular, a user can define a special "class", a user-defined blueprint with a set of properties or functions that are common to all objects of this datatype. The user could then create different objects as "instances" of this datatype, just as a user might create different variables to be instances of the `Integer` datatype. Additionally, these classes can then be "extended" to define new datatypes. These new datatypes "inherit" the attributes and methods of their parent class, but may also possess new attributes and methods of their own.

The framework uses the `Node` class to define the `Agent` and `Network` classes. The `Agent` and `Network` classes can be further extended by the modeller to define the individual agents and locations of their own model. The `Agent` class possesses a `addBehaviour` function which can be used to define agent behaviours.

In Scala, `case` classes are particularly useful for modeling immutable data. In the models specified below, the agents and network locations are thus defined as `case` classes that extend these inbuilt framework-defined classes. Additionally, BharatSim implements a Finite State Machine (FSM); with this implementation, a modeller may define classes that extend the inbuilt `State` trait to distinguish agents that exist in different disease states. In order to use this FSM, the framework extends the `Agent` class to define `StatefulAgent`, an agent that can be in only one `State` at any given instant of time. The `State` class possesses two inbuilt functions:

a. An `enterAction`: a function that defines an action that is to be performed only whenever an agent enters this particular state, and

b. A `perTickAction`: a function that defines an action that is repeated by an agent at every time-step (or "tick") for as long as they are within this state.

Additionally, every extension of the `State` class also possesses an `addTransition` function which allows the modeller to define (i) which `State` a `StatefulAgent` can transition to from the current `State`, and (ii) under what conditions such a transition can occur. Very much like the `addBehaviour` function, the `addTransition` function is run for every agent at every time-step, to check whether a transition between states should occur.

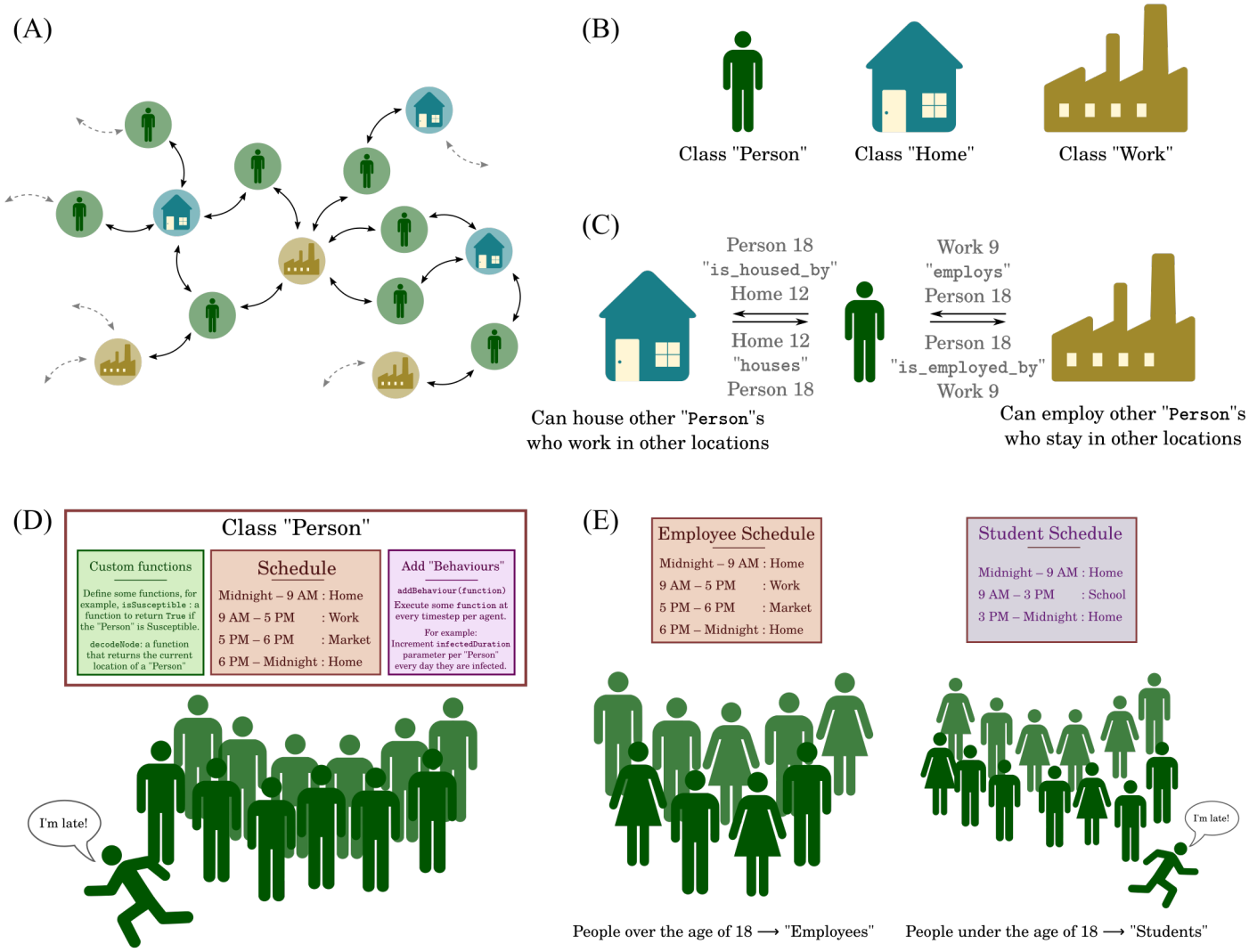

Fig 4. Components of the simulation engine. (A) A schematic of a graph between nodes representing agents, homes, and work locations. (B) These nodes are user-defined extensions of the in-built `Node` class. (C) The graph is created by creating bi-directional relationships between these user-defined classes. (D) Agents can have user-defined schedules using user-defined units of time. Additionally, the agent class may posses custom user-defined functions. Some of these function – called "behaviours" – can be set so that they are executed for every agent at every time-step. (E) Agents' schedules can be customized based on attributes from the synthetic population like their age, job label, and socioeconomic status.

Agents are assumed to move between different network locations to interact with each other. These locations can be modelled by extending the inbuilt `Network` class. Just as with the `Agent` class, these classes too may possess an arbitrary number of attributes. Additionally, the `Network` class has an inbuilt `getContactProbability` function which can be used to define, for example, a location dependent relative risk of infection.

Every individual agent follows a schedule that is defined by the modeller. Such schedules specify agent locations across time. These schedules can be dynamic, can depend on the current state of the agent, and can be affected by interventions that are

imposed. In addition to the predefined units of time (`Hour`, `Day`, `Week`, etc.) schedules may also be linked to user-defined units of time (instances of the `ScheduleUnit` class). In BharatSim, these schedules can be registered using the `registerSchedules` function. When schedules are registered, a priority can also be assigned to them. This may be used to give a lockdown-schedule a higher priority than the routine daily schedule of any agent.

2.2.2 Defining disease dynamics across `States`

The movement of individual agents between different states can be modelled using either transitions or behaviours, depending on whether we use the inbuilt finite state machine or not. If an individual agent is susceptible, then we decide whether or not they may become infected using a probability that depends on the number of infected individuals they are in contact with in a given interval of time.

When an agent transitions into any other disease state, an "exit-time" is chosen, based on the current time and the probability distribution associated with that disease state. The modeller can use the inbuilt random number generators in *Scala* to define and use these generators. If an exiting agent has multiple possible states to transition into, one of these states is chosen based on the disease parameters. The individual then stays in this disease state until the current time exceeds their exit-time, at which point a further transition occurs.

2.2.3 Running a simulation

For a typical simulation (see Appendix S6) we define a `Person` class as an extension of a `StatefulAgent`, in addition to two extensions of the `Network` class: `Home` which represents the agent's household, and `Office`, which represents their workplace. The disease states of our model are defined as extensions of the `State` trait.

Once these basic components are defined, we create the `Main` class of our simulation. Within this, we create a new instance of the framework's `Simulation` class. The synthetic population is then imported using the `ingestData` function of the `Simulation` class. This is done using the framework-defined `ingestCSVData` function. This function accepts a CSV file as well as a user-defined "mapping" function which creates a graph. At the end of the population import, a graph is created, containing all the individual agents and their attributes, as well as all the home and work locations in our simulation.

During the population import process, relations are established between the different nodes. Relations are single-directional links between two nodes, distinguished by a string label. For example, a `Person` and `Home` are linked by the `"STAYS_AT"` relation. Similarly, a `Home` `"HOUSES"` a `Person`. Relations can be established between any two nodes, for example two `Home` nodes can be related using a `"NEIGHBOUR"` relation, and so on.

Once the synthetic population has been imported and the graph created, we call the `defineSimulation` function of the `Simulation` class. Within this, we (i) register the agents in our simulation using `registerAgent` function, and (ii) register the different disease states in our simulation using the `registerState` function (if we are using the inbuilt FSM).

We can then define schedules that the individual agents are required to follow, based on their attributes (age, employment status, and so on). Once the schedules have been defined, we can also register an action to be performed based on user-defined conditions given to the `registerAction` function. For example, one could choose to call the predefined `StopSimulation` action (which ends the simulation) when the number of infected agents in the population is zero.

The modeller can also create a new instance of the `CsvOutputGenerator` class and send it to the `register` function of the `SimulationListenerRegistry`. This function is called at every time step and uses the `CsvOutputGenerator` to write a single file to an output file. This can be used to store, for example, the aggregate numbers of susceptible, infected, and recovered individuals in our population. The modeller can use the `onCompleteSimulation` function of the `Simulation` to run specific commands when the entire simulation has finished. This function should also end with the `teardown` function which clears the data and closes the connection with the database.

Further details, including a step-by-step guide of how to create a simple agent-based SIR model using BharatSim, can be found in the "Writing your first program" section of the BharatSim documentation (Appendix S6).

2.3 The visualization engine

Using the simulation engine, the modeller can write simulation data to an output file. Such a file could contain, for example, the number of individuals in different disease states for every time step, or more specific geo-spatial information. The visualization engine can then read this output and create multiple dashboards with different types of graphs that can help to analyze the results and visualize them.

The visualisation engine is self-contained, and can accept and visualize any CSV data file. Each dashboard provides the user with a combination of different graphs and charts, including line-graphs, histograms, and pie-charts. Additionally, the visualization engine can also represent GIS data in the geoJSON format to plot heatmaps and choropleths. These can either be static, or can change with time based on the data provided in the input CSV file.

The visualization engine provides the following features:

Data Import The visualization engine can import data in ZIP, CSV, and GeoJSON formats. The user can add, delete, and link files from different dashboards and can also add columns with custom formulae.

Charts The user can represent time-series data as a line-graph. Additionally, bar charts, histograms, and pie charts can be used to study aggregated data. Geographical information and data can also be visualized. Spatial variations in data can be visualized using either heatmaps or choropleths. Heatmaps visualize data in the form of "hot" or "cold" spots, with a warm-to-cool color scheme. Choropleth maps can also be used to visualize how a quantities vary across a fixed geographic area like a district or state, while simultaneously showing the extent of variation within a region.

Project & Dashboard management Users can create a Project which can have multiple individual dashboards. Users can add, edit, and delete projects and dashboards for better management. Dashboards are auto-saved to avoid data loss. Users can easily add, edit, duplicate, and delete widgets and charts in each dashboards. Dashboards can be duplicated to allow users to duplicate certain design choices and widget configurations. Each widget also allows the users to export the output either as raster or vector data using the PNG and SVG file formats respectively.

2.4 Implementing a disease progression

The BharatSim framework allows a modeller to implement a variety of agent-based models. The disease states in these models can be labelled according to the compartments in associated compartmental models. In this paper, we restrict ourselves

to the compartments described in the INDSCI-SIM model of Ref [47], although this may be easily generalized to other models. This model uses the disease progression shown in Fig 5. It has been used to describe the disease trajectory across the first wave of COVID-19 in India. Extensions of the INDSCI-SIM model that incorporate vaccinations have been used to describe subsequent waves in the country [48].

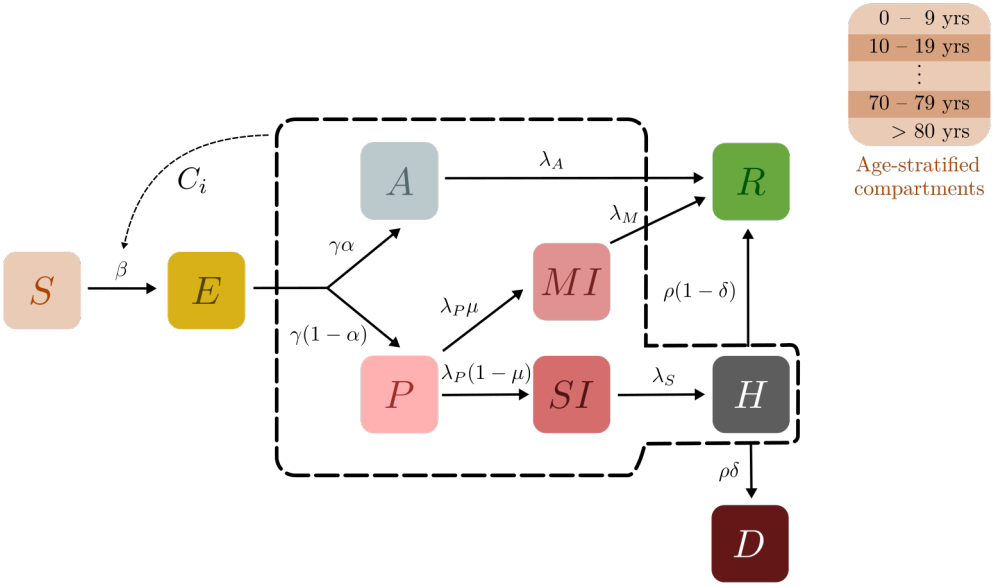

Fig 5. Schematic of the epidemiological model. Boxes show the 9 compartments of the model, i.e., the set of states that characterise each individual in the population. The compartments are (S)usceptible, (E)xposed, (A)symptomatic, (P)resymptomatic, Mildly Infected (MI), Severely Infected (SI), (H)ospitalized, (R)ecovered, and (D)ead. Black arrows indicate possible transitions between states. The dotted lines indicate that the rate of infection of Susceptible individuals is an increasing function of the number of infected individuals they are in contact with. Recovered individuals are assumed to be immune to further infection in this model, however the possibility of including a non-zero recovery rate has been implemented.

The INDSCI-SIM model assumes 9 compartments: Susceptible (S), Exposed (E), Asymptomatic (A), Presymptomatic (P), Mildly Infected (MI), Severely Infected (SI), Hospitalized (H), Recovered (R), and Dead (D). Those in the Susceptible compartment can be infected by all individuals in any infectious compartment, whereupon they transition to the exposed compartment. Exposed individuals can either become asymptomatic, exhibiting no symptoms and eventually recovering, or presymptomatic, wherein they transition into either the mildly or severely infected compartments. Mildly infected individuals recover, while severely infected ones are eventually hospitalized. They go on to either recover or die.

The synthetic population described in Section 2.1 provides us with individual-level information about the age distribution, household size distribution and composition, and comorbidities. This information provides further granularity to the description of disease states and the transitions between them. In the model discussed here, for example, the individual agent's probability of branching between the asymptomatic and presymptomatic states is age-dependent, with older individuals more likely to develop an acute case of the illness. We do the same for the other branching probabilities, as shown in Table 4.

The residence times that different individuals spend in different compartments can, in principle, also be age-stratified. For simplicity here, we assume these parameters to be the same across all age-groups. BharatSim allows the modeller the possibility of choosing these residence times from different probability distributions including the exponential distribution, as simple compartmental models assume, or the lognormal distribution which we use in this paper on grounds of its wider applicability. A set of convenient parameters for these different distributions can be obtained from [49], and can be found in Table 3. These can be updated as more information becomes available.

Parameter	Description	Distribution
τ_E	Exposed-to-infectious duration	Lognormal(4.5,1.5)
τ_{sym}	Time to onset of symptoms	Lognormal(1.1,0.9)
τ_A	Recovery time for Asymptomatics	Lognormal(8,2)
τ_{MI}	Recovery time for Mild cases	Lognormal(8,2)
τ_{SI}	Time before hospitalization	Lognormal(1.5,2)
τ_H	Time spent hospitalized	Lognormal(18.1,6.3)

Table 3. Residence times in different compartments. When an individual agent enters a disease state, they remain in this compartment for a period of time drawn from the probability distributions given above [49].

Parameter	0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	80-89	90+
Relative Susceptibility	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Asymptomatic Probability	0.5	0.45	0.40	0.35	0.30	0.25	0.20	0.15	0.10	0.10
Severe Infection Probability	0.0005	0.00165	0.00720	0.02080	0.03430	0.07650	0.13280	0.20655	0.24570	0.24570
Probability of Death	0.00002	0.00002	0.00010	0.00032	0.00098	0.00265	0.00766	0.02439	0.08292	0.16190

Table 4. Branching ratios for different compartments. In our model, we assume that an individual agent's probability of branching between different compartments is age-stratified. Individuals are divided into age-bands of one decade each. When an individual agent exits a compartment that branches into multiple possible compartments, they are assigned a compartment based on this age-stratified probability.

We also incorporate the possibility of vaccination drives in our model. Vaccines are assumed to reduce both the relative risk of infection as well as the probability of symptomatic disease. We also assume that vaccinated individuals are less likely to transmit the disease, as shown in [50]. The vaccine's efficacy is assumed to ramp-up to its maximum value over a period of 14 days. Additionally, the dose-interval is currently set to 90 days between doses; see Appendix S4 for more information.

All individuals above the age of 20 are assumed to be eligible to receive vaccines, provided they are not exhibiting symptoms. We can begin our simulation with either a vaccine-naive population, or a population where a certain fraction has already received partial protection through one or two shots of the vaccine. For the simulations discussed here, these numbers are chosen to be 20% and 10% respectively, to match the situation in India as of August 1, 2021. These initial vaccines are distributed as follows: first, 80% of the individuals above the age of 60 are assumed to be vaccinated by the start of the simulation. Next, any remaining vaccines are distributed randomly among the full (age-stratified) eligible population.

Additionally, vaccines can be rolled out in different phases, targeting eligible individuals of different age-groups. In the simulations described below, we include two phases. In the first phase, individuals above the age of 40 are given first preference to receive vaccines. This phase spans one month. In case there are no eligible agents above the age of 40, the vaccines are randomly distributed among the entire population to avoid allotment-related wastage. In the second phase, all agents above the age of 20 (not currently exhibiting symptoms) are targeted for vaccination.

2.5 Network dynamics

An individual's movements in a day are defined through a "schedule" (see Section 2.2), specific to each individual. This schedule dictates which network location (home or workplace) the individual participates in at any given time. The infection spreads stochastically between individuals who share a location at any given time, at rates determined by the infection parameters as well as the vaccination status of the different individuals present in that location. In our simulations, we take a single time step to be 12 hours, defining the probabilities accordingly. Individuals move back and forth between their homes and workplaces or schools twice a day.

Schools are a specific subset of workplaces to which all school-going individuals are assigned. Schools consist of classrooms, and students are assigned specific classrooms and do not mix with students in other classrooms. Multiple network and school structures can be examined. For example, one may allow for students and/or teachers to move between the classrooms of the school. Classrooms may also be defined according to age brackets.

The force of infection $\mathcal{F}$ at any location l, the rate at which a single susceptible person becomes infected, is proportional to the product of the number of infectious people at that location and the infectivity parameter λ_S. This is divided by a normalization factor $\mathcal{V}_l$:

$$\mathcal{F}(l) = \frac{\lambda_S}{\mathcal{V}_l} \sum_{\substack{\text{agent } i \\ \text{in } l}} V_{\text{tr}}(i)\ C_q(i), \tag{1}$$

where $V_{\text{tr}}(i)$ is the factor by which the agent i's probability of transmission is changed due to vaccination, $C_q(i)$ is an "isolation factor" (chosen in the range 0^+ to 1). In our simulations below, this factor is 0.1 if the agent is hospitalized or quarantined and 1 if not. For density-dependent transmission, as in this paper, we can chose $\mathcal{V}_l$ to be the effective number of individuals in location l, not counting dead individuals and only a fraction of hospitalized individuals, as in [51]. However, this can easily be generalized to other types of transmission, including frequency-dependent transmission.

The force of infection at each location controls the probability p_j that a susceptible individual j in that location contracts the disease and therefore transitions to the exposed state in some time-step Δt:

$$p_j = V_{\text{inf}}(j) \times \mathcal{F}(l) \times \Delta t, \tag{2}$$

where $V_{\text{inf}}(j)$ is the relative risk of infection, given by the vaccination status of the individual j.

When an individual enters the exposed compartment or any of the subsequent compartments, a "residence-time" is drawn and stored from the distributions shown in Table 3. The individual remains in this compartment for this period. Once this time has elapsed, the individual transitions to the next state in the disease progression. If two possible outcomes exist – for example, suppose they can transition to both the asymptomatic or presymptomatic states – then one of the outcomes is chosen randomly,

weighted by the appropriate (age-stratified) branching probability for that transition. For the new state, a new "sojourn-time" is drawn and stored.

Following an update of all agents, the time is incremented by Δt, all individuals are moved along the network according to their specified schedules, and the process repeated for the next time-step.

2.6 The models and underlying paradigms

There are two ways to model the disease progression in BharatSim and the modeller may choose between then. Both work with the same disease progression and synthetic population and implement the same interventions. In the first, the framework's inbuilt Finite-State Machine (FSM) controls the movement of agents between the different disease states. The framework ensures that an individual `StatefulAgent` can only possess one `activeState` at any given instant of time. Such an implementation is computationally efficient and simple to debug.

When we wish to model situations where multiple strains of a disease could coexist, such an FSM cannot be implemented without a proliferation of states. Indeed, an individual could have recovered from one strain, only to be infected with another, or might in principle be simultaneously infected with both strains. In the second implementation, we do not use the `StatefulAgent` class, but instead work with the `Agent` class. In this case, the individual's state is defined by a set of two attributes: `s1_disease_state` and `s2_disease_state`, which contain information of which states they are currently in with respect to the disease dynamics of each individual strain. In the model described here, being infected by one strain reduces an individual's relative risk of infection to the second strain. We vary this relative protection from the second strain to see how varying levels of immunity provided by the first strain change the disease trajectory. (Reinfections for each individual strain are currently not implemented in the examples discussed below, although it is easy to include them.)

2.7 Interventions and complex dynamics

2.7.1 Lockdowns and school-closures

Non-pharmaceutical interventions like lockdowns or school-closures can also be implemented in BharatSim. For a simulated lockdown, individual agents' schedules are modified so that they remain at home. All agents are assumed to follow this rule, except those that are designated as essential-workers. A tunable fraction may additionally have a low adherence to policy-level interventions. School-closures work in a similar fashion: all students and teachers assigned to a school stay at home, depending on their level of adherence.

2.7.2 Vaccination strategies

A vaccination drive can be modelled as one of many types of `Interventions` in the BharatSim framework. Here, we extend the INDSCI-SIM model to incorporate a two-shot vaccination drive [48,52]. We allow for a fixed daily vaccination rate as some fraction of the total population (note that this is not a fraction of the eligible population). Thus, with a total population of 10,000, a daily vaccination rate of 0.2% would mean that there are 20 vaccines that can be distributed among the eligible population every day. This number can be varied, so as to compare different allocations of vaccine doses corresponding to 0.0, 0.2, 0.4, and 0.8% of the population. Peak vaccination levels in India corresponded to roughly 0.4%. Because our model is age-structured, we can prioritize different age groups [53], and also allow for a fraction of the population to be vaccinated at the start of the simulation.

Vaccines are modeled through adjusting individuals agents' susceptibility to infection (V_{inf}) and probability of developing symptoms after being infected (which is controlled by the age-stratified branching parameter λ_A, the probability of an individual being asymptomatic) [54]. Both of these changes cause a reduction in the overall probability of severe disease, hospitalization, and death. Vaccines are assumed to also reduce transmissibility by a factor V_{tr}. In principle, disease severity could be a function of other features captured in the synthetic population, such as individual co-morbidities. For simplicity, we ignore this possibility here.

Our models further allow us to implement a two-dose vaccination strategy [55], with dose prioritization: the maximum protection that the first-dose provides is attained through a linear ramp-up over some period T_{ramp}. Once this value is attained, the parameters that vaccination affects remain unchanged for some "inter-dose" interval T_{id} until the vaccinated individual becomes eligible for (and receives) a second dose, following which their protection is increased through another linear ramp-up.

Over the course of the simulation, vaccinations are administered at the end of every day. Dose prioritization is done by first allocating a certain fraction of the vaccines as "second-dose" vaccines. These will be used to vaccinate any agents eligible for a second dose on that particular day. Any remaining doses are distributed as first doses. To make this concrete consider a vaccine drive that begins on day 0, with daily vaccination rate of 1%, and an inter-dose interval of 90 days. We further assume a dose-prioritization of 80:20, that is, 20% of the vaccines are set assigned for second-doses while the remainder are used for first-doses. For the first 90 days, no individuals are eligible for second-doses. As a result, all the vaccines are administered to individuals requiring first-doses. On day 90, the first 10 individuals who received their first dose on day 0 are now eligible for second-dose vaccination. Thus, of the 20 vaccines set aside, 10 are used to get them fully vaccinated. The remaining 10 (as well as the assigned 80) are used as first-dose vaccines. In a similar vein, if any vaccines that are set aside for first-doses and unused are used as second-doses where possible. This procedure ensures that no vaccines are wasted while eligible people remain in the population. See Appendix S5 for more information.

In this paper, for illustrative purposes, we have chosen a vaccination drive that is rolled-out in two phases. For the first 30 days, individuals older than 40 years of age are prioritised for vaccines. After this period, vaccines are provided to all individuals over the age of 20. Any excess vaccines are then distributed randomly among the remaining adult population, when possible, thus ensuring no wastage. We also ensure that 80% of the initial vaccinated population consists of individuals above the age of 60 years. We set V_{tr} (the reduction in transmissibility) to be 40%. The linear ramp-up period over which the maximum vaccine efficacy is attained and the minimum duration between doses are set to $T_{\text{ramp}} = 14$ days and $T_{\text{id}} = 90$ days, respectively. The doses are prioritized in a ratio of 80:20, with 20% of the doses being set aside as second-doses, whenever eligible agents are available.

2.7.3 The impact of school-reopenings

We consider circumstances under which governments may consider allowing schools to be reopened in the background of a pandemic. To do this, we construct communities that simulate the catchment area for a given school. This community is chosen to be a section of the synthetic population consisting of 20,316 individuals and with 6500 homes, 120 workplaces, and 1 school. Of the individual-level attributes included in the synthetic population, we restrict ourselves to the age, and the home, work, and school locations of the individuals.

In the school, individuals are assigned to a specific classroom, and no movement is allowed between classrooms. Thus, in our simulations the relevant parameter is the number (and consequently the density) in the classrooms. We assume 100 classrooms

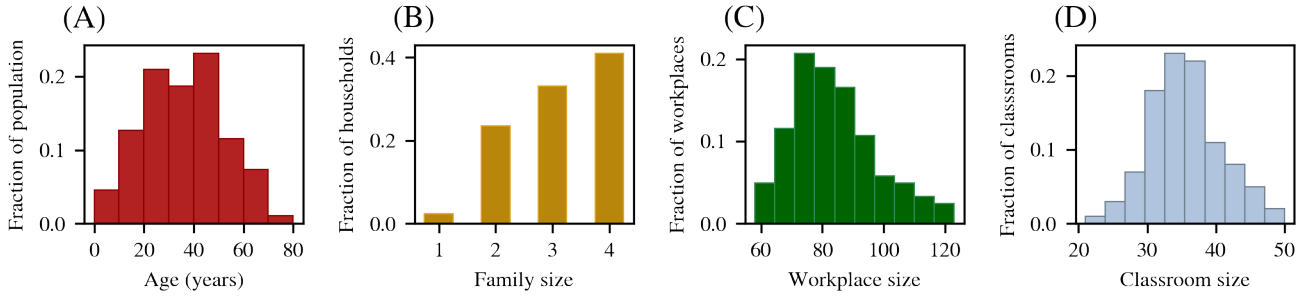

Fig 6. School synthetic population statistics. A section of the synthetic population for the city of Pune is chosen with 20,316 individuals with 6500 homes, 120 workplaces, and 1 school. These individuals represent the catchment area for the school. We plot the distributions of (A) age, (B) family size, (C) workplace size, and (D) classroom size for this section of the population.

per school. Fig 6 shows the distribution of home sizes, workplace sizes, and classroom sizes in our model community, along with the age distribution. A certain fraction of the individuals in the population (30% in our case) is homebound.

2.7.4 Introducing multiple strains

Infections with different variants follow the same disease progression indicated in Fig 5, with different transition parameters. Here we assume that the two strains differ only in their value of λ_S. The first strain is considered to be less transmissible, i.e. with a lower value of λ_S, than the second one. All other parameters are identical, including the residence times for compartments and the branching ratios.

In addition, we consider the possibility of the first less-transmissible strain providing an agent with some immunity to infection from the second strain [56]. It is thus more likely that the second strain will infect someone who has escaped infection from the first strain. We vary this likelihood, as well as levels of vaccination [57], to explore counterfactual scenarios. Additionally, we also assume that an agent who has contracted the first strain is more likely to be an asymptomatic carrier of second strain than a symptomatic one, and vice versa. The second (more transmissible) strain is seeded with 10 individuals after the first strain has nearly ended. However, our simulation allows for both strains to be seeded simultaneously and to coexist as well [58–60].

3 Results

3.1 Simulating disease spread in a model Indian city

We use our model to simulate the city of Pune, with a population of 3.13 million agents, exploring the different disease trajectories induced by pharmaceutical and non-pharmaceutical interventions.

We consider first the effect of a lockdown on the disease trajectory. Fig 7A shows the fraction of recovered over all age-groups with time in different scenarios. In these scenarios, the daily vaccination rate is varied, and 15-day lockdowns are introduced when the total number of infections crosses a threshold of either 1% of 5% of the total population. If left unchecked the pandemic would have led to 70% of the population having contracted the disease during the epidemic. With the introduction of a 15-day lockdown that begins when the number of active cases is 5% of the total population, this number reduces to close to 60%. The number further reduces if the lockdown began even earlier (when the number of active cases is 1% of the population).

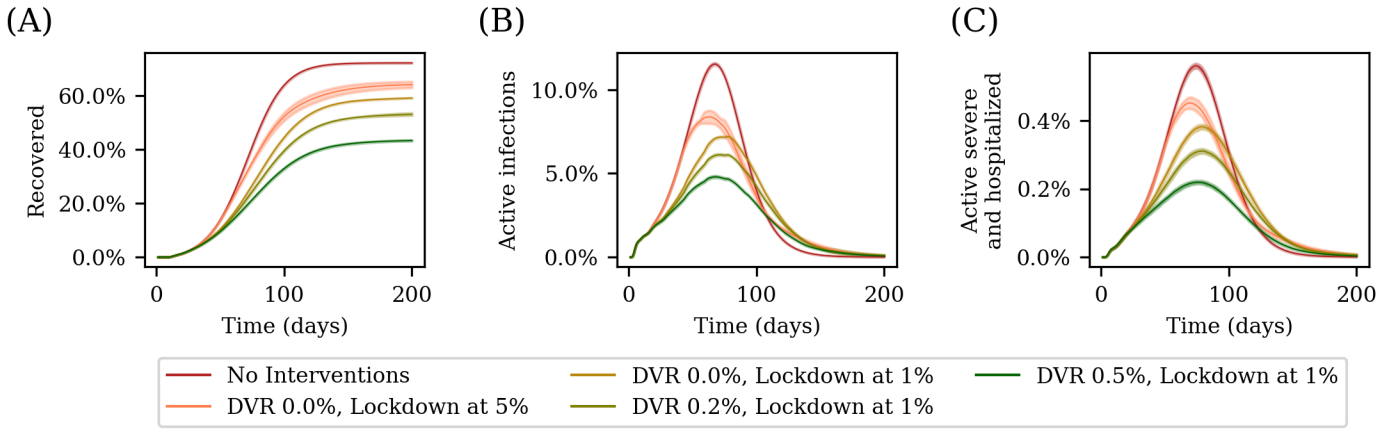

Fig 7. Simulating disease trajectories with different interventions. The plots show the disease trajectory for the entire city of Pune in five different scenarios: without any interventions, with 15-day lockdowns beginning when the number of active infections is 1% and 5% of the population respectively, with a lockdown starting with 1% active infections and concurrent vaccination drive at two different vaccination rates of 0.2% and 0.5%. (A) shows the number of recovered as a function of time, (B) the number of active cases as a function of time, and (C) the number of active severe cases and hospitalizations. In all scenarios with a vaccination drive, the drive begins on day 0. The curves are averages over 10 runs, with the error bars corresponding to 1.96σ.

We next consider the role of vaccinations. We begin a lockdown when the number of active infections is 1% of the population, and vary the daily vaccination rate of the vaccination drive. By comparing these scenarios with the curve representing the case with the lockdown but no vaccination drive in Fig 7A and Fig 7B, we see that an increased daily vaccination rate reduces peak of the infection from from 10.8% of the total population when there was no vaccination to 7.4% and 4.4% in the case of vaccination drives with 0.2% and 0.5% daily vaccination rates respectively. The number of severe and hospitalized, shown in Fig 7C, follows the active-infections curve, although lagging slightly behind it in all scenarios.

Additionally, we also study what effect the initial spatial configuration of the disease has on its trajectory. In Fig 8, we compare two such scenarios: in the first, a total of 3000 individuals (roughly 0.1% of the total population of Pune) are seeded in a single ward, and the disease is allowed to spread. In the second scenario, the same number of individuals are chosen from a much smaller geographical region (a collection of 750 homes), and the disease is again allowed to spread. The figure compares the spread of the disease due to these two initial conditions.

Finally, we also compare the effects of different lockdown strategies on disease dynamics. In particular, we study the effect that locking down the most populous wards of the city has on the spread of the disease. In order to do this, the following procedure is used: the entire city is divided into a grid whose unit cell is 200 m $\times$ 200 m. Every five days, we count the number of infected individuals in each of these grid boxes. An individual is assumed to count towards the number of infected in a grid box if their household falls within its geographical boundaries. We place the most populous central wards under a lockdown for (i) 15 days and (ii) 30 days when the number of cases is 1% of the total population, prohibiting all entry and exit from these wards, and observe the effect this has on the disease dynamics.

In reality, the south-west of Pune saw a larger number of cases than other regions. This has been attributed to the relative importance of public transportation, variations in adherence to non-pharmaceutical interventions and inhomogeneities in case-records,

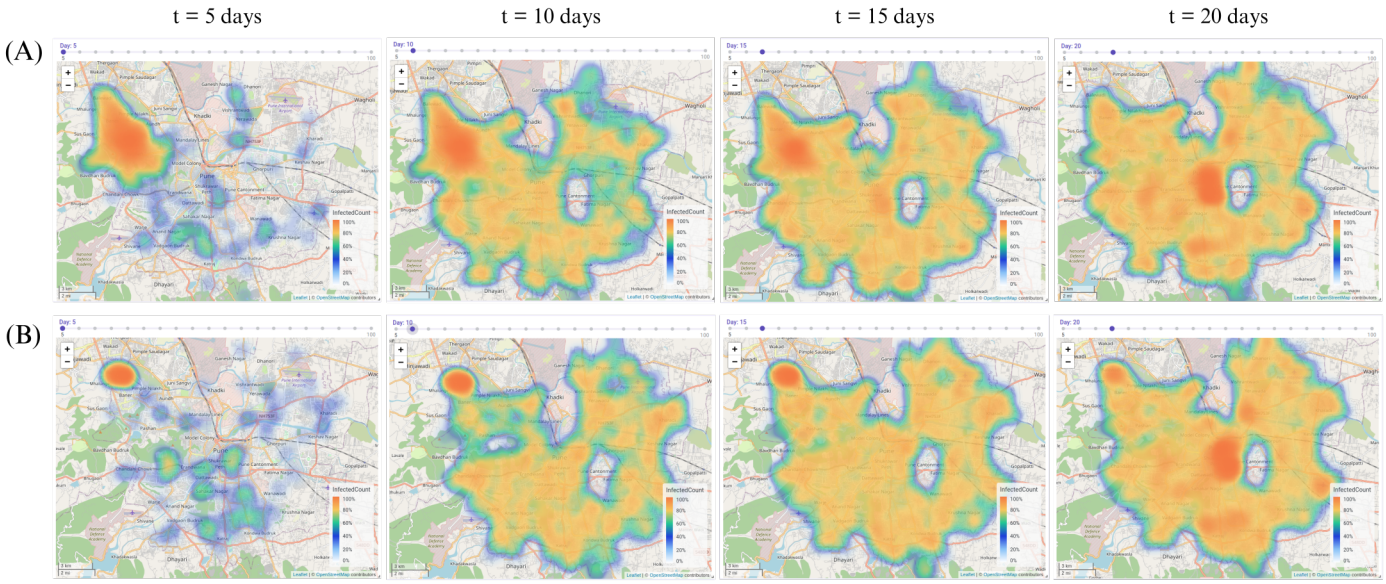

Fig 8. Effect of initial infection seed on disease spread. The heatmaps represent the density of the infected at different instants of time for two scenarios. In each case, the initial infection seed of 0.1% of the total population was seeded in (A) a single ward, and (B) 750 families in a single ward. The central ellipse that remains unchanged across time represents the Army Cantonment region for which no publicly-available data exists.

among other explanations. Our model here is not intended to precisely reproduce these complexities, but to describe a framework within which such questions can be addressed.

3.2 Effects of school reopening on the disease trajectory

To study the impact of school reopening on the disease trajectory, we simulate a number of different counterfactual scenarios. In these simulations, we assume schools are initially closed (i.e., children stay at home) as was the case in India for a close-to two-year period since March 2020. We then examine the effect of opening schools at various times during the epidemic. We assume 20% of our population is initially vaccinated, with 10% having received the second-dose. We further assume that vaccinations proceed at a constant daily vaccination rate with an appropriately chosen prioritization of first and second doses.

We consider two different initial conditions. In the first, 30% of the population is assumed to have already recovered from the disease, and is therefore immune to it. In the second case, this number is set to 50% [61]. In both cases, we observe the effect of varying vaccination rates (including no vaccination), as well as the times at which schools are reopened.

Fig 10 shows our results for infected children (ages 0-18) and adults (ages 18+) as a function of time. We find that for children, reopening schools could lead to a substantial increase in cases with an initial recovered fraction of 30%. In this case, opening schools earlier leads to a greater increase in cases among children. However, with an initial recovered fraction of 50%, this increase is substantially lower. In all simulations with these initial conditions, reopening schools does not lead to any significant increase in infections among adults. As a consequence, the total number of severe infections, hospitalizations, and deaths remains largely unaffected when schools are reopened under any of these scenarios.

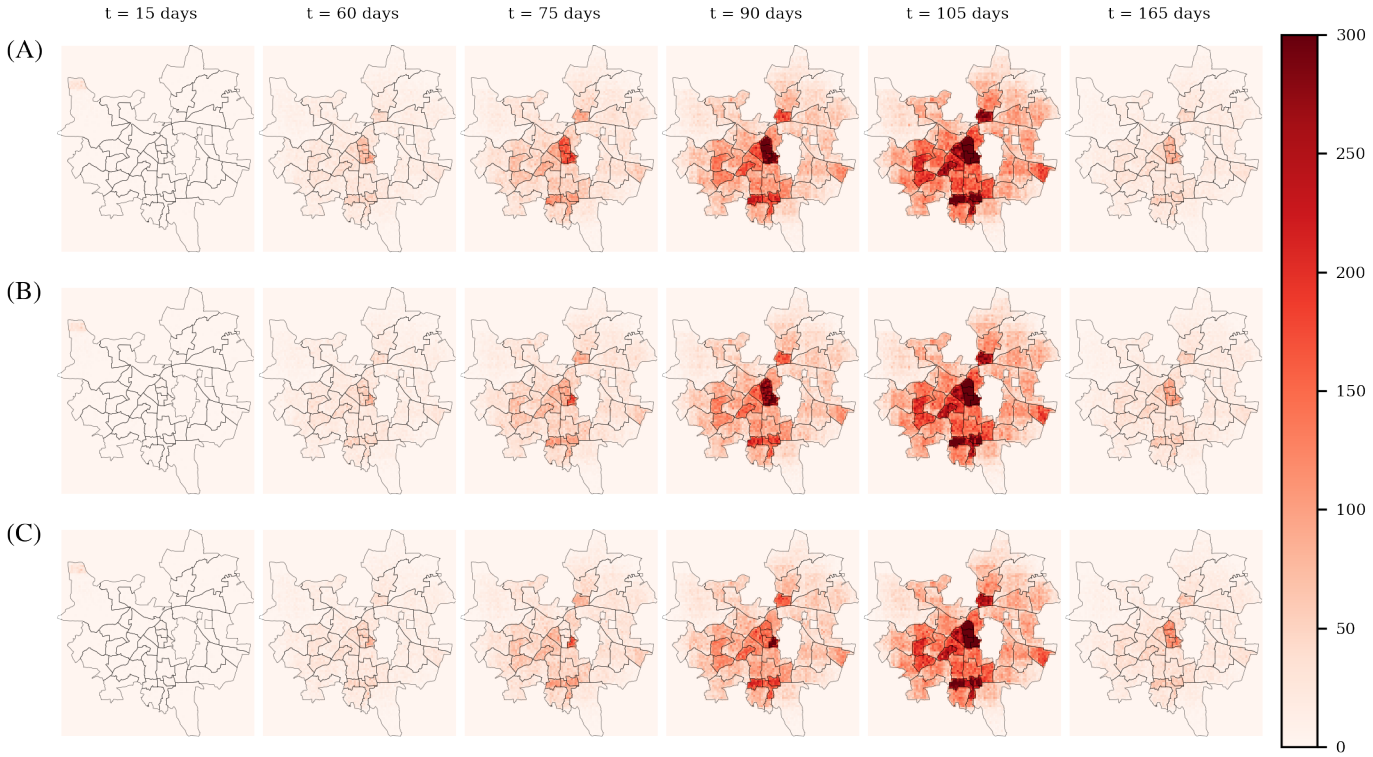

Fig 9. Effects of different lockdown strategies. The city of Pune is divided into a grid of 200 m × 200 m, and the number of infected individuals with households in each grid box is plotted as a heatmap. In scenario (A) no interventions are applied and the disease is allowed to progress freely through the population. In order to test the effect that locking down the most populous wards has, these wards are locked down for a period of (B) 15 days (C) and 30 days when the number of active infections is 1% of the total population, which corresponds to day 50. As before, the central region that remains unchanged represents the Army Cantonment area for which we have no data.

However, any increase of cases among children, even if asymptomatic or mild, may also be a matter of concern. Thus, we also studied the peak number of cases in children as a function of the time at which schools are reopened. This is shown in Fig 11 for initial recovered fractions of 30% and 50%. From these plots, we see that after a certain time, keeping schools closed makes only a marginal difference to the peak number of infections amongst children. This threshold time depends on both the daily vaccination rate and the initial seropositivity. In all cases, this threshold lies well past the date of peak of the infection that would have occurred if schools had been closed throughout. Thus, if cases in children have already reduced significantly from the peak value, reopening schools would only make a marginal difference.

Another useful criterion for this threshold is obtained by examining the vaccine coverage and net seropositivity as a function of time (see Table 5). Here we define vaccine coverage as the fraction of the total population (i.e., both adults and children) that has received at least one dose. We define net seropositivity as the fraction of the total population that has either been infected and recovered, or has been vaccinated by at least one dose. For this to hold true, the net infection-induced seropositivity among children must be comparable to that among adults.

Comparing Fig 11 with Table 5, we conclude that in order to minimize the peak

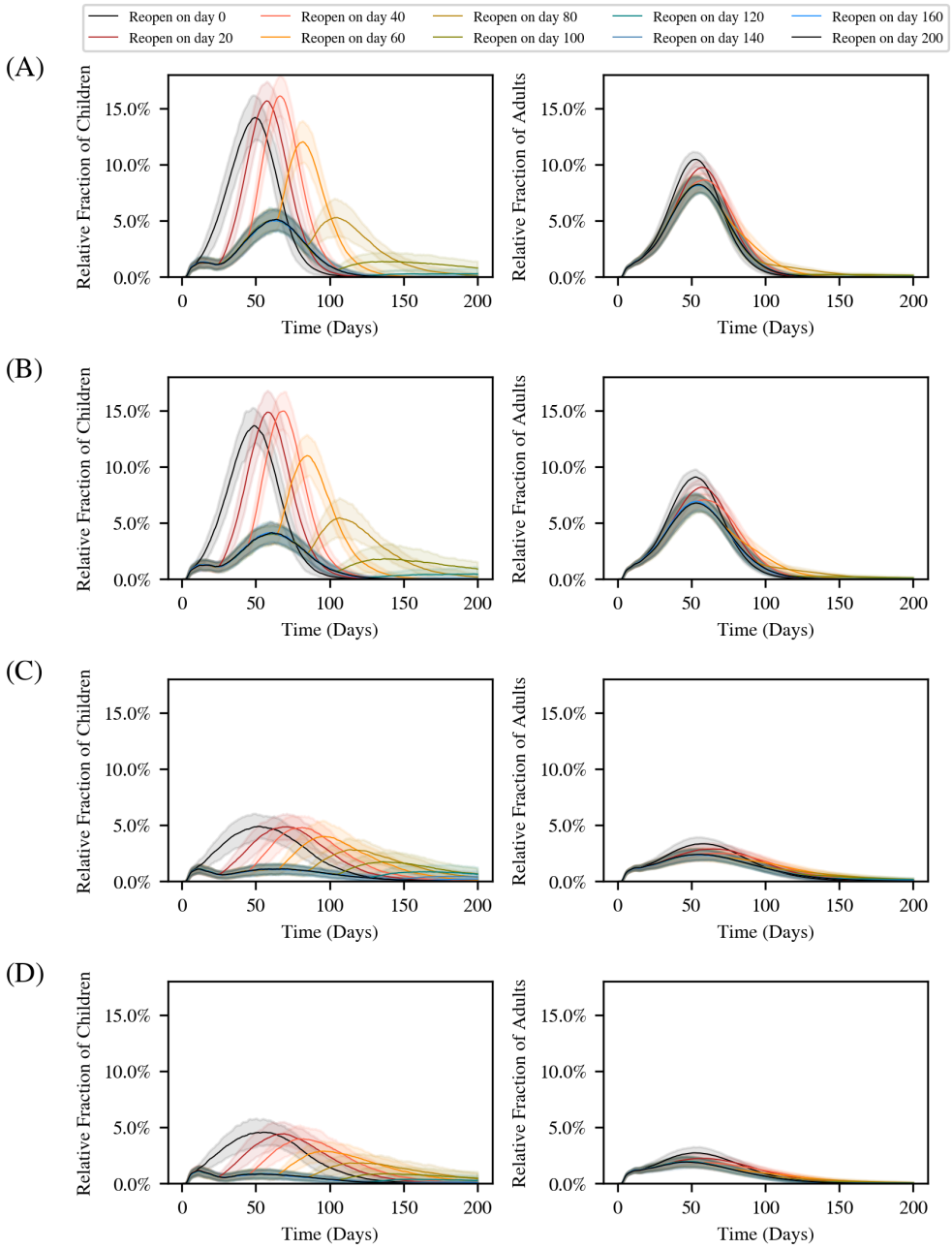

Fig 10. Simulating different school reopening timelines. The left-panels show results for children (≤ 18 years) while the right-panels show the results for adults (> 18 years). We begin with an initial recovered fraction of 30% of the population, and with 20% of the population vaccinated (10% with the first dose and 10% with both doses). We compare two different scenarios: in (A) we simulate the population in the absence of a vaccination drive, and in (B) with a daily vaccination rate of 0.4%. The initial plateau in active cases for children is due to the fact that a small fraction of children are infected as part of the initial infection seed, but are not able to infect each other since schools are closed. In (C) and (D) we compare the same scenarios (no vaccination drive and 0.4% daily vaccination rate respectively), but with an initial recovered fraction of 50%. In all cases, opening schools earlier leads to a rise in cases among children, but no significant increase in infections among adults. The curves are all averaged over 100 simulation runs, with the error bars corresponding to 1.96σ.

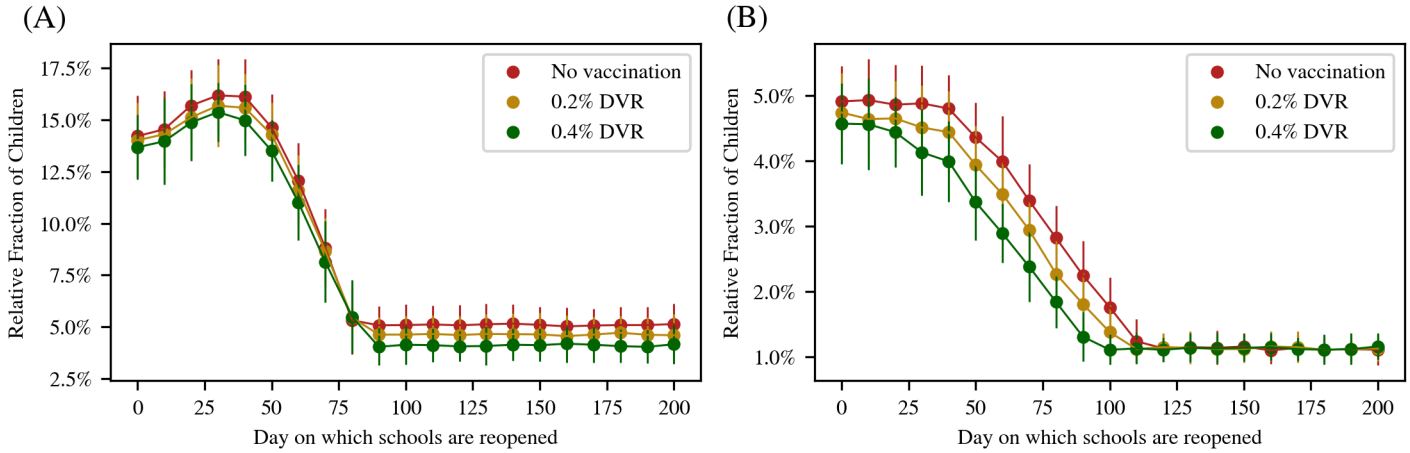

Fig 11. Effect of schools reopening on the epidemic peak in children. The peak number of children infected (normalized to their relative population) under different reopening times as indicated by the $x-$axis and different rates of immunization as indicated by the color of the plots. The two panels show the results (A) assuming 30% initial recovered (B) assuming 50% initial recovered. The values are all averaged over 100 simulation runs, with the error bars corresponding to 1.96σ.

DVR	Unlock Schools on Day	0	20	40	60	80	100	120	140	160	180	200
0%	**Vaccine Coverage**	20	20	20	20	20	20	20	20	20	20	20
	Net Seroprevalence	44.8	48.0	58.2	71.3	77.9	79.6	79.9	80.0	80.0	80.0	80.0
0.2%	**Vaccine Coverage**	20	23	26.1	29.3	32.4	35.6	38.7	41.9	45.0	48.2	51.3
	Net Seroprevalence	44.8	50	61	73.3	79.6	81.9	82.9	83.7	84.5	85.2	85.9
0.4%	**Vaccine Coverage**	20	26.1	32.4	38.8	45.2	51.6	58.0	64.3	70.7	77.1	83.5
	Net Seroprevalence	44.8	51.9	63.8	75.5	81.8	84.7	86.6	88.2	89.9	91.5	93.1

DVR	Unlock Schools on Day	0	20	40	60	80	100	120	140	160	180	200
0%	**Vaccine Coverage**	20	20	20	20	20	20	20	20	20	20	20
	Net Seroprevalence	60.8	62.7	66.2	70.1	73.2	75.3	76.2	76.8	77	77.2	77.2
0.2%	**Vaccine Coverage**	20	23	26.1	29.3	32.4	35.6	38.7	41.9	45	48.2	51.3
	Net Seroprevalence	60.8	64.1	68.7	72.9	76.5	78.5	80.3	81.3	82.4	83.4	84.3
0.4%	**Vaccine Coverage**	20	26.1	32.4	38.8	45.2	51.6	58.0	64.3	70.7	77.1	83.5
	Net Seroprevalence	60.8	65.5	71.1	76.1	79.8	82.8	85.1	87.2	89.3	91.3	93.3

Table 5. Vaccine coverage and net seropositivity when schools are reopened. The fraction of the population that have received at least one vaccine dose (vaccine coverage) and the net seropositivity (both infection and vaccination derived) are tabulated at the moment schools are reopened, for simulations with daily vaccination rates (DVR) of 0% (no vaccination), 0.2% and 0.4%. The top table shows results for an initial recovered fraction of 30% and the bottom corresponds to an initial recovered fraction of 50%.

number of infections in children, schools should be unlocked only when the net seropositivity reaches the range 75-85%. This net seropositivity will be achieved by a combination of natural infections and vaccination. Therefore, vaccinating the population at a higher rate will allow one to open earlier, as will an initially higher

seropositivity. Our model allows us to compute these thresholds for any arbitrary vaccination schedule and prior history of infections.

3.3 Modeling multiple strains with variable immunity

Finally, we study how the trajectory of a disease is influenced by the introduction of a new and more transmissible strain. The two strains are modelled as identical, except for the parameter β. We wish to examine how levels of immunity granted by both infection- and vaccination-induced immunity alter the disease trajectory of the second strain.

We consider a community of 10,000 unvaccinated individuals. We initially seed the population with 10 individuals infected by the first strain. The infection is then allowed to propagate without intervention. Once the infections due to this strain have died out, we then introduce the more transmissible second strain by seeding 100 individuals at random.

Individuals who recover from infections with the first strain have their disease trajectories following a second infection altered in two ways: (i) they have a lower relative risk of infection when they come in contact with infected individuals (modeled by introducing a "reinfection probability" parameter), and (ii) prior infection reduces an individual's risk of severe illness, with a protection equal to a single vaccine dose [48].

The trajectory of the second-wave of the disease, induced by the more transmissible variant, is shown in Fig 12. The worst case occurs when the probability of reinfection is 100%. In this case, since the variant is more transmissible and can infect the entire population, the peak in infections is substantially higher than that of the first-wave. As the reinfection probability is reduced, this peak is diminished.

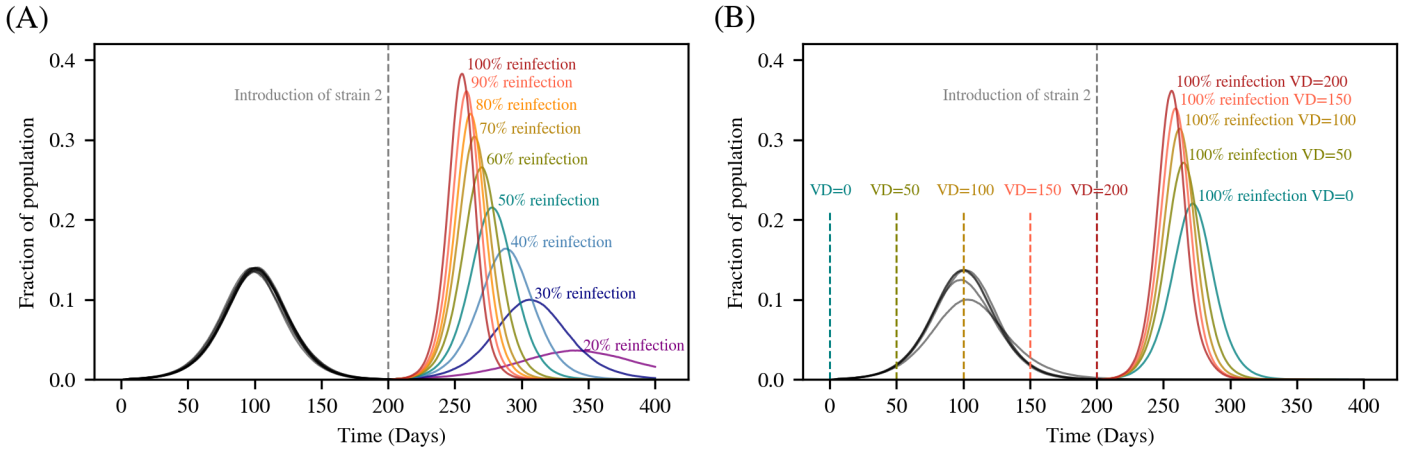

Fig 12. Effect of reinfection probability on disease progression. A second more virulent strain is introduced once the first strain has died out. Individuals who contracted the first strain have a reduced relative risk of infection to the second strain, representing a "reinfection probability". In (A) the reinfection probability is varied, producing different disease trajectories. As the reinfection probability reduces, so does the peak of the second wave. In (B) we study the effect of a delayed vaccination drive on the disease trajectory. A constant daily vaccination rate of 0.5% is used. For simplicity, only the curve for 100% reinfection is shown, although the trend is similar for other values of the reinfection probabilities. Starting the vaccination drive later causes the more virulent strain to spread much faster, reaching a much higher peak earlier.

We can further study the effect that vaccination has on the disease trajectory by delaying the start of the vaccination drive. In Fig 12A, the drive is assumed to start at the start of the simulation, but we can modify this. Fig 12B shows what happens when

the start of the vaccination drive is delayed by successive intervals of 50 days. For simplicity, only the curve for 100% reinfection is shown.

As can be expected, delaying the start of the vaccination drive causes the second, more virulent, strain to spread much faster, and reach a higher peak earlier. The trend is identical for lower daily vaccination rates, although the relative difference between the peaks is less noticeable.

4 Discussion

This paper has described the construction and applications of BharatSim, an agent-based simulation framework designed for India. The design of BharatSim includes the construction of a detailed synthetic population based on survey and census data for India, a simulation framework that incorporates this population, as well as a tool that can be used to easily visualize the simulations. Spatially resolved individual-level models like BharatSim provide a powerful way of describing the spread of an infectious disease but are not restricted to this context. They are particularly useful for discussions of policy, because insights into social behavior and into how individuals make decisions can be incorporated into them. The complexity of the underlying dynamical contact networks is also easily incorporated into models of this type.

We applied BharatSim to the study of three specific questions pertaining to the spread of COVID-19 in India: (i) How do local and global measures (eg. lockdowns and vaccination drives) affect the disease trajectory of the epidemic in a model Indian city? (ii) When is it feasible to reopen schools during an epidemic? (iii) How does partial immunity arising from a previous infection or vaccination affect the trajectory of a new immune-evasive variant? All of these questions have attracted attention from the point of view of COVID-19 policy. BharatSim provides an independent way of assessing multiple counterfactual scenarios in which these questions may be addressed.

We found that the time of initiation and the vaccination rate associated with a vaccination drive can be tuned so as to have the maximum effect on the disease trajectory. Additionally, we established that at the levels of seropositivity reported across much of India in the wake of the first and second waves of COVID-19, schools could have been reopened with relatively little impact several months before they were actually opened. Finally, we found that the effects of a vaccination drive, if timed appropriately, could blunt the increase of cases due to a second, more transmissible, viral strain.

BharatSim, although applied only to COVID-19 in the scenarios described here, lends itself to more versatile use. Describing the direct transmission of any other disease transmitted through proximity is straightforward. With the addition of components describing environmental variables, BharatSim can be extended to describe, say, the transmission and spread of typhoid within communities. This would require a parametrization both of hydrology and water-use practices, tied to geographical variables. Adding in the life cycle and dynamics of arthropod vectors such as mosquitoes to the dynamics of infected agents allows the modelling of vector-borne diseases such as malaria and dengue [62–65]. Mosquitoes could also be modelled as agents [66, 67], but a more practical description might be in terms of dynamical fields which interact with, and are modified by, the presence of human agents [66, 68]. Indeed, beginning with assumptions about individual behavior and social determinants, coupled to the synthetic population of interest, one could potentially describe the incidence of non-communicable diseases in time for a prescribed population, calibrating these results against available data.

Agent-based models are powerful because the behaviour of large groups of individual actors can be emergent as a consequence of their interactions. Thus, BharatSim should

find use in the social sciences, where computational methods are increasingly being utilized to overcome restrictions of standard economic and game-theoretic modeling [69]. The methodology of agent-based models is particularly useful in problems when agents can be programmed to choose from a range of behaviors in response to exogenous information. These may derive from their interactions with a limited number of other agents or from environmental variables. The tendency of sub-groups of individuals within a society to interact largely within themselves, for example groups of individuals bound together through social or economic ties or caste affiliations, can be described in terms of the interaction networks that define the synthetic population. Thus, frameworks such as BharatSim provide us with a way of potentially assessing the social determinants of disease, through information, usable within models, about how individuals make decisions while embedded in their social and economic contexts. We note too that while BharatSim comes with a synthetic population that is India-specific, nothing prevents the use of the simulation and visualization engines on a similarly constructed population for any part of the world.

A particularly powerful use of BharatSim might be the following: India has initiated the single largest digital public-health initiative in the world, the Ayushman Bharat Digital Mission (ABDM) [70]. Its broad aim is the development of an integrated digital health infrastructure. The ABDM will create a national digital health ecosystem that supports Universal Health Coverage. Data, information and services integrated under the ABDM will be accessible via an open, interoperable digital system. Every Indian will have a digital health ID, the ABHA number. A comprehensive repository of all health facilities in India across different systems of medicine, including public and private health facilities, will be completed in parallel.

The debate around the ABDM has centred around two main questions: of privacy and, separately, of the digital divide. Concerns with privacy associated with the direct use of ABDM data could potentially be alleviated if these data were to be made the basis for high-quality synthetic populations. Once such a population is constructed and rigorously benchmarked, running scenarios that can be tweaked in real-time becomes possible. Regarding the digital divide, one could hope to use these methods to identify populations that are under-served, potentially correlating these with social determinants of health that would need to be addressed in parallel.

These possibilities for further investigation point to the usefulness of methods such as BharatSim to interrogate specific policy interventions in health and in those areas that overlap it. These could draw from what we know to be the social and economic determinants of health in India and similar LMICs. To our knowledge, such areas of investigation remain relatively unexplored, particularly in the complex social environment that India presents.

Acknowledgments

GIM and PC are grateful to Brian Wahl, Sandeep Krishna, Farhina Mozaffer, Sandeep Juneja, Harish Iyer, Kayla Laserson, Anand Krishnan, Rajesh Sundaresan, Siva Athreya, Mihir Arjunwadkar, and Joy Monteiro for many discussions. The authors from Ashoka University are especially grateful for support from the Mphasis F1 Foundation. GIM and PC acknowledge support from the World Health Organization SAGE Working Group on Vaccines (APW Contract #202706833). GIM acknowledges support of the Bill and Melinda Gates Foundation, Grant No: R/BMG/PHY/GMN/20 and the Tata Institute of Fundamental Research for an Adjunct Professorship, while PC acknowledges support from Ashoka University and the Mphasis F1 Foundation. The authors acknowledge the use of computational facilities provided by the Centre for Bioinformatics and Computational Biology at Ashoka University. Yerik Singh

contributed to the online documentation, in particular the section on the Simulation Engine, and we thank him for his help. The Thoughtworks team would like to acknowledge the sponsors of the Engineering for Research initiative for supporting the development of the *EpiRust* simulator. The funders had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

1. Bernoulli D, d'Alembert JLR. Smallpox inoculation: an eighteenth century mathematical controversy. Matlock: University of Nottingham, Department of Adult Education; 1971.

2. Ross R. The prevention of malaria. New York: Dutton; 1910. Available from: https://catalog.hathitrust.org/Record/001587831.

3. Hamer SWH. Epidemiology, Old and New. The Anglo-French Library of Medical and Biological Science. Broadway House, 68-74 Carter Lane, London E.C.: Kegan Paul, Trench, Trubner & Co., Ltd.; 1928.

4. Kermack WO, McKendrick AG, Walker GT. A contribution to the mathematical theory of epidemics. Proceedings of the Royal Society of London Series A, Containing Papers of a Mathematical and Physical Character. 1927;115(772):700–721. doi:10.1098/rspa.1927.0118.

5. Anderson RM, May RM, Anderson B. Infectious Diseases of Humans: Dynamics and Control. Revised ed. edition ed. Oxford: Oxford University Press; 1992.

6. Keeling MJ, Rohani P. Modeling Infectious Diseases in Humans and Animals. Illustrated edition ed. Princeton: Princeton University Press; 2007.

7. Syme SL. Social determinants of disease. Annals of Clinical Research. 1987;19:44–52.

8. Adler NE, Stewart J. Preface to the biology of disadvantage: socioeconomic status and health. Annals of the New York Academy of Sciences. 2010;1186:1–4. doi:10.1111/j.1749-6632.2009.05385.x.

9. Marmot M, Bell R. Fair society, healthy lives. Public Health. 2012;126 Suppl 1:S4–S10. doi:10.1016/j.puhe.2012.05.014.

10. Braveman P, Gottlieb L. The Social Determinants of Health: It's Time to Consider the Causes of the Causes. Public Health Reports. 2014;129(Suppl 2):19–31.

11. Islam MM. Social Determinants of Health and Related Inequalities: Confusion and Implications. Frontiers in Public Health. 2019;7:11. doi:10.3389/fpubh.2019.00011.

12. Lee H, Kim D, Lee S, Fawcett J. The concepts of health inequality, disparities and equity in the era of population health. Applied nursing research: ANR. 2020;56:151367. doi:10.1016/j.apnr.2020.151367.

13. Luke DA, Stamatakis KA. Systems science methods in public health: Dynamics, networks, and agents. Annual Review of Public Health. 2012;33(1):357–376. doi:10.1146/annurev-publhealth-031210-101222.

14. Macal CM, North MJ. Tutorial on agent-based modeling and simulation. In: Proceedings of the Winter Simulation Conference, 2005.; 2005. p. 14 pp.–.

15. Bonabeau E. Agent-Based Modeling: Methods and Techniques for Simulating Human Systems. Proceedings of the National Academy of Sciences of the United States of America. 2002;99(10):7280–7287.

16. Hunter E, Mac Namee B, Kelleher JD. A Taxonomy for Agent-Based Models in Human Infectious Disease Epidemiology. Journal of Artificial Societies and Social Simulation. 2017;20(3):2.

17. Barrett CL, Bisset KR, Eubank SG, Feng X, Marathe MV. EpiSimdemics: An efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In: SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing; 2008. p. 1–12.

18. Eubank S, Guclu H, Anil Kumar VS, Marathe MV, Srinivasan A, Toroczkai Z, et al. Modelling disease outbreaks in realistic urban social networks. Nature. 2004;429(6988):180–184. doi:10.1038/nature02541.

19. Hoops S, Chen J, Adiga A, Lewis B, Mortveit H, Baek H, et al. High Performance Agent-Based Modeling to Study Realistic Contact Tracing Protocols. In: 2021 Winter Simulation Conference (WSC); 2021. p. 1–12.

20. Islam KA, Marathe M, Mortveit H, Swarup S, Vullikanti A. Data-driven Agent-based Models for Optimal Evacuation of Large Metropolitan Areas for Improved Disaster Planning. Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. 2022; p. 1639–1641.

21. Dickens BL, Koo JR, Lim JT, Park M, Quaye S, Sun H, et al. Modelling lockdown and exit strategies for COVID-19 in Singapore. The Lancet Regional Health - Western Pacific. 2020;1:100004. doi:10.1016/j.lanwpc.2020.100004.

22. Guerstein S, Romeo-Aznar V, Dekel M, Miron O, Davidovitch N, Puzis R, et al. The interplay between vaccination and social distancing strategies affects COVID19 population-level outcomes. PLOS Computational Biology. 2021;17(8):e1009319. doi:10.1371/journal.pcbi.1009319.

23. Bhattacharya P, Chen J, Hoops S, Machi D, Lewis B, Venkatramanan S, et al. Data-driven scalable pipeline using national agent-based models for real-time pandemic response and decision support. The International Journal of High Performance Computing Applications. 2023;37(1):4–27. doi:10.1177/10943420221127034.

24. Halloran ME, Longini IM, Nizam A, Yang Y. Containing Bioterrorist Smallpox. Science. 2002;298(5597):1428–1432. doi:10.1126/science.1074674.

25. Ferguson NM, Cummings DAT, Cauchemez S, Fraser C, Riley S, Meeyai A, et al. Strategies for containing an emerging influenza pandemic in Southeast Asia. Nature. 2005;437(7056):209–214. doi:10.1038/nature04017.

26. Ferguson NM, Cummings DAT, Fraser C, Cajka JC, Cooley PC, Burke DS. Strategies for mitigating an influenza pandemic. Nature. 2006;442(7101):448–452. doi:10.1038/nature04795.

27. Germann TC, Kadau K, Longini IM, Macken CA. Mitigation strategies for pandemic influenza in the United States. Proceedings of the National Academy of Sciences. 2006;103(15):5935–5940. doi:10.1073/pnas.0601266103.

28. Chao DL, Halloran ME, Obenchain VJ, Jr IML. FluTE, a Publicly Available Stochastic Influenza Epidemic Simulation Model. PLOS Computational Biology. 2010;6(1):e1000656. doi:10.1371/journal.pcbi.1000656.

29. Athreya S, Menon GI, Sundaresan R. COVID-19 modeling for India and a roadmap for the future. Communications of the ACM. 2022;65(11):82–87. doi:10.1145/3557798.

30. Babu GR, Ray D, Bhaduri R, Halder A, Kundu R, Menon GI, et al. COVID-19 Pandemic in India: Through the Lens of Modeling. Global Health: Science and Practice. 2021;9(2):220–228. doi:10.9745/GHSP-D-21-00233.

31. Agrawal S, Bhandari S, Bhattacharjee A, Deo A, Dixit NM, Harsha P, et al. City-Scale Agent-Based Simulators for the Study of Non-pharmaceutical Interventions in the Context of the COVID-19 Epidemic. Journal of the Indian Institute of Science. 2020;100(4):809–847. doi:10.1007/s41745-020-00211-3.

32. Kshirsagar JK, Dewan A, Hayatnagarkar HG. EpiRust: Towards A framework for large-scale agent-based epidemiological simulations using rust language. In: Linkoping Electronic Conference Proceedings. vol. 176:67. Linkoping Electronic Conference Proceedings; 2021. p. 475–482. Available from: https://doi.org/10.3384/ecp20176475.

33. Kale S, Bawaji S, Dewan A, Dhanani M, Gupta K, Hayatnagarkar H, et al. Parallel and distributed epirust: Towards billion-scale agent-based epidemic simulations. In: Proceedings of the 15th international conference on agents and artificial intelligence - volume 1: ICAART,. SciTePress; 2023. p. 297–307.

34. Hazra DK, Pujari BS, Shekatkar SM, Mozaffer F, Sinha S, Guttal V, et al. The INDSCI-SIM model for COVID-19 in India. medRxiv; 2021. Available from: https://www.medrxiv.org/content/10.1101/2021.06.02.21258203v1.

35. Neekhra B, Kapoor K, Gupta D. Synthpop++: A hybrid framework for generating A country-scale synthetic population; 2023.

36. Desai S, Vanneman R, National Council of Applied Economic Research. India human development survey-II (IHDS-II), 2011-12. Inter-university Consortium for Political and Social Research; 2018.

37. Office of the Census Commissioner of India. Census tables; 2011. Available from: https://censusindia.gov.in/census.website/data/census-tables.

38. National Sample Survey Office, NSSO. NSS 68th round; 2012. Available from: http://www.icssrdataservice.in/datarepository/index.php/catalog/91.

39. CIESIN - Columbia University. Gridded population of the world, (GPWv4): Population density; 2016.

40. Ye X, Konduri KC, Pendyala RM, Sana B, Waddell P. Methodology to Match Distributions of Both Household and Person Attributes in Generation of Synthetic Populations. In: 88th Annual Meeting of the Transportation Research Board; 2009.Available from: https://trid.trb.org/view/881554.

41. Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K. Modeling tabular data using conditional GAN. In: Wallach H, Larochelle H, Beygelzimer A, dAlché Buc F, Fox E, Garnett R, editors. Advances in neural information processing systems. vol. 32. Curran Associates, Inc.; 2019.Available from:

https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

42. Ministry of Education G. Steps taken by government to provide education to poor student; 2019. Available from: https://pib.gov.in/PressReleasePage.aspx?PRID=1578389.

43. Shalev-Shwartz S, Ben-David S. Understanding Machine Learning: From Theory to Algorithms. 1st ed. Cambridge University Press; 2014.

44. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.

45. Bishop CM. Pattern Recognition and Machine Learning. Softcover reprint of the original 1st ed. 2006 edition ed. New York, NY: Springer; 2016.

46. The Scala Programming Language;. Available from: https://www.scala-lang.org/.

47. Hazra DK, Pujari BS, Shekatkar SM, Mozaffer F, Sinha S, Guttal V, et al. Modelling the first wave of COVID-19 in India. PLOS Computational Biology. 2022;18(10):e1010632. doi:10.1371/journal.pcbi.1010632.

48. Mozaffer F, Cherian P, Krishna S, Wahl B, Menon GI. Effect of hybrid immunity, school reopening, and the Omicron variant on the trajectory of the COVID-19 epidemic in India: a modelling study. The Lancet Regional Health - Southeast Asia. 2023;8:100095. doi:10.1016/j.lansea.2022.100095.

49. Kerr CC, Stuart RM, Mistry D, Abeysuriya RG, Rosenfeld K, Hart GR, et al. Covasim: An agent-based model of COVID-19 dynamics and interventions. PLOS Computational Biology. 2021;17(7):e1009149. doi:10.1371/journal.pcbi.1009149.

50. Lucas C, Vogels CBF, Yildirim I, Rothman JE, Lu P, Monteiro V, et al. Impact of circulating SARS-CoV-2 variants on mRNA vaccine-induced immunity. Nature. 2021;600(7889):523–529. doi:10.1038/s41586-021-04085-y.

51. Cherian P, Krishna S, Menon GI. Optimizing testing for COVID-19 in India. PLOS Computational Biology. 2021;17(7):e1009126. doi:10.1371/journal.pcbi.1009126.

52. Mandal S, Arinaminpathy N, Bhargava B, Panda S. Responsive and agile vaccination strategies against COVID-19 in India. The Lancet Global Health. 2021;9(9):e1197–e1200. doi:10.1016/S2214-109X(21)00284-9.

53. Saadi N, Chi YL, Ghosh S, Eggo RM, McCarthy CV, Quaife M, et al. Models of COVID-19 vaccine prioritisation: a systematic literature search and narrative review. BMC Medicine. 2021;19(1):318. doi:10.1186/s12916-021-02190-3.

54. Ella R, Reddy S, Jogdand H, Sarangi V, Ganneru B, Prasad S, et al. Safety and immunogenicity of an inactivated SARS-CoV-2 vaccine, BBV152: interim results from a double-blind, randomised, multicentre, phase 2 trial, and 3-month follow-up of a double-blind, randomised phase 1 trial. The Lancet Infectious Diseases. 2021;21(7):950–961. doi:10.1016/S1473-3099(21)00070-0.

55. Kumar VM, Pandi-Perumal SR, Trakht I, Thyagarajan SP. Strategy for COVID-19 vaccination in India: the country with the second highest population and number of cases. npj Vaccines. 2021;6(1):1–7. doi:10.1038/s41541-021-00327-2.

56. Singh J, Anantharaj A, Panwar A, Rani C, Bhardwaj M, Kumar P, et al. BA.1, BA.2 and BA.2.75 variants show comparable replication kinetics, reduced impact on epithelial barrier and elicit cross-neutralizing antibodies. PLOS Pathogens. 2023;19(2):e1011196. doi:10.1371/journal.ppat.1011196.

57. Castro e Silva A, Bernardes AT, Barbosa EAG, Chagas IASd, Dáttilo W, Reis AB, et al. Successive Pandemic Waves with Different Virulent Strains and the Effects of Vaccination for SARS-CoV-2. Vaccines. 2022;10(3):343. doi:10.3390/vaccines10030343.

58. Pattabiraman C, Habib F, K HP, Rasheed R, Prasad P, Reddy V, et al. Genomic epidemiology reveals multiple introductions and spread of SARS-CoV-2 in the Indian state of Karnataka. PLOS ONE. 2020;15(12):e0243412. doi:10.1371/journal.pone.0243412.

59. Thacker T. 'All Omicron strains present in India, but in small numbers'. The Economic Times. 2022;.

60. Sreenivas D, Kumar V, Kathirvel K, Vadnala RN, Mishra S, Shelar B, et al.. Genomic surveillance reveals circulation of multiple variants and lineages of SARS-CoV-2 during COVID-19 pandemic in Indian city of Bengaluru; 2023. Available from: https://www.biorxiv.org/content/10.1101/2023.03.14.532352v1.

61. Padma MR, Dinesh P, Sundaresan R, Athreya S, Shiju S, Maroor PS, et al. Second round statewide sentinel-based population survey for estimation of the burden of active infection and anti-SARS-CoV-2 IgG antibodies in the general population of Karnataka, India, during January-February 2021. IJID Regions. 2021;1:107–116. doi:10.1016/j.ijregi.2021.10.008.

62. Takahashi LT, Maidana NA, Ferreira WC, Pulino P, Yang HM. Mathematical models for the Aedes aegypti dispersal dynamics: travelling waves by wing and wind. Bulletin of Mathematical Biology. 2005;67(3):509–528. doi:10.1016/j.bulm.2004.08.005.

63. Maidana NA, Yang HM. Describing the geographic spread of dengue disease by traveling waves. Mathematical Biosciences. 2008;215(1):64–77. doi:10.1016/j.mbs.2008.05.008.

64. Yamashita WMS, Takahashi LT, Chapiro G. Dispersive models describing mosquitoes' population dynamics. Journal of Physics: Conference Series. 2016;738(1):012065. doi:10.1088/1742-6596/738/1/012065.

65. Magori K, Legros M, Puente ME, Focks DA, Scott TW, Lloyd AL, et al. Skeeter Buster: A Stochastic, Spatially Explicit Modeling Tool for Studying Aedes aegypti Population Replacement and Population Suppression Strategies. PLOS Neglected Tropical Diseases. 2009;3(9):e508. doi:10.1371/journal.pntd.0000508.

66. Smith NR, Trauer JM, Gambhir M, Richards JS, Maude RJ, Keith JM, et al. Agent-based models of malaria transmission: a systematic review. Malaria Journal. 2018;17(1):299. doi:10.1186/s12936-018-2442-y.

67. Pizzitutti F, Pan W, Feingold B, Zaitchik B, Álvarez CA, Mena CF. Out of the net: An agent-based model to study human movements influence on local-scale malaria transmission. PLOS ONE. 2018;13(3):e0193493. doi:10.1371/journal.pone.0193493.

68. Romeo-Aznar V, Paul R, Telle O, Pascual M. Mosquito-borne transmission in urban landscapes: the missing link between vector abundance and human density. Proceedings of the Royal Society B: Biological Sciences. 2018;285(1884):20180826. doi:10.1098/rspb.2018.0826.

69. Fischbach K, Marx J, Weitzel T. Agent-based modeling in social sciences. Journal of Business Economics. 2021;91(9):1263–1270. doi:10.1007/s11573-021-01070-9.

70. NHA | Official website Ayushman Bharat Digital Mission; 2023. Available from: https://abdm.gov.in/.

Supporting information

Appendix S1 Scaling BharatSim with population size and model complexity. The simulation times for different population sizes are compared, both for a simple SIR model and the more complicated INDSCI-SIM model described here. We show that the BharatSim simulation engine can handle populations of close to 10 million agents, and that the simulation time scales up as $\mathcal{O}(N)$.

Appendix S2 Benchmarking the synthetic population. We benchmark the quality of the synthetic population by comparing the statistics of the synthetic population and survey data using a chi-squared test for categorical data (sex) and a Kolmogorov-Smirnov test for continuous data (age, height, and weight). In each case, the confidence that the synthetic data and the survey data come from the same distribution is reported.

Appendix S3 Description of the compartmental model. The disease-progression in a single well-mixed compartmental model is briefly described.

Appendix S4 The effect of vaccination. We show how vaccination affects both the relative risks of infection, and of contracting severe disease. These risks are governed by modifying individual agents' β and α parameters respectively. When an agent receives a dose of the vaccine, their protection increases linearly over a 14 day window, until it plateaus to the maximum protection offered by that dose. The reduction in the relative risk of infection (i.e. in β) is assumed to be the same for each age-group, while the reduction in the risk of contracting severe disease is age-stratified.

Appendix S5 Vaccine dose-prioritization and avoiding wastage. The vaccine doses are distributed between first- and second-shot vaccinations in an 80:20 ratio, in such a way as to minimize dose-wastage, as discussed in the main text. We discuss the algorithm used to enforce this, and illustrate through examples how the doses are distributed in a population of 10,000 agents for different daily vaccination rates.

Appendix S6 Getting started with BharatSim. The BharatSim documentation, available from bharatsim.readthedocs.io, comes with a short tutorial on Writing your First Program which introduces the basic concepts of the simulation framework through instructions on how to write a simple SIR model using BharatSim.