SOFTWARE

The Kendrick Modelling Platform: Language Abstractions and Tools for Epidemiology

Mai Anh BUI T.[1*](anhbtm@soict.hust.edu.vn) , Nick Papoulias[2](npapoylias@gmail.com) , Serge Stinckwich[3,4,5](serge.stinckwich@ird.fr) , Mikal Ziane[6,7](mikal.ziane@lip6.fr) and Benjamin Roche[2](benjamin.roche@ird.fr)

[*]Correspondence:
anhbtm@soict.hust.edu.vn
[1]Software Engineering Department, School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
Full list of author information is available at the end of the article

Abstract

Background: Mathematical and computational models are widely used for examining transmission, pathogenicity and propagation of infectious diseases. Software implementation of such models and their accompanied modelling tools do not adhere to well established software engineering principles. These principles include both *modularity* and *clear separation of concerns* that can promote *reproducibility* and *reusability* by other researchers. On the contrary the software written for epidemiology is monolithic, highly coupled and severely heterogeneous. This reality ultimately makes these computational models hard to study and to reuse, both because of the programming competence required and because of the incompatibility of the different approaches involved. Our goal with **Kendrick** is to simplify the creation of epidemiological models through a unified Domain-Specific Language for epidemiology that can support a variety of modelling and simulation approaches classically used in the field. This goal can be achieved by promoting *reproducibility* and *reuse* with modular modelling abstractions.

Results: We show through several examples how our modular abstractions and tools can reproduce uniformly complex mathematical and computational models of epidemics, despite being simulated by different methods. This is achieved without requiring sophisticated programming skills from the part of the user. We then successfully validate each kind of simulation through statistical analysis between the time series generated and the known theoretical expectations.

Conclusions: **Kendrick** is one of the few DSLs for epidemiology that does not burden its users with implementation details or expecting sophisticated programming skills. It is also currently the only language for epidemiology that supports modularity through clear separation of concerns that promote reproducibility and reuse. **Kendrick**'s wider adoption and further development from the epidemiological community could boost research productivity in epidemiology by allowing researchers to easily reproduce and reuse each other's software models and simulations. The tool can also be used by people who are not necessarily epidemiology modelers.

Keywords: Domain-specific language; Modularity; Mathematical modelling; Epidemiological modelling; Compartmental models

Background

During the last decades, mathematical and computational modelling became widely used for investigating the mechanisms of infectious diseases propagation [1, 2], ex-

ploring their evolutionary dynamics [3, 4], and/or informing control strategies [5, 6]. Their insights are increasingly recognised since the early 1980s when human population entered a period of emergence and re-emergence of infectious diseases [7, 8]. Including the increasing number of public health threats, such as a possible smallpox epidemic due to bioterrorism attack [9], emergence of pandemic influenza [10] or the need to propose new kinds of vaccination programs [11]. Indeed, the complexity of these new infections, relying on many inter-connected factors [12, 13, 14] such as biodiversity decline [15], antibiotics resistance [7] or intensification of worldwide trade [16], makes the anticipation of their propagation and their evolution a very tricky challenge against which epidemiological models can be crucial tools.

Epidemiological models largely rely on the so-called SIR framework [17, 18] where host population is divided into categories corresponding to their epidemiological status. Basically, epidemiological models consider individuals who are Susceptible to the pathogen (state S) and then can be infected, Infectious (state I) that can transmit the disease and Recovered (state R) who are immunised and cannot become infected again. Such model aims to characterise the transition between these categories and the consequences on the dynamics of each category, generally the 'Infectious' one that contains diseased individuals. From this initial configuration, an infinity of other categories can be added in order to represent different transmission cycles corresponding to each infectious disease that could be studied.

Based on this framework, an epidemiological model can nevertheless be implemented and simulated in different ways. Generally, the first approach is to define the life cycle through a system of ordinary differential equations in order to be analytically tractable or to be deterministically simulated through numerical methods, such as Runge-Kutta [19]. While this approach is especially useful to understand the average dynamics without chance, shifting to a stochastic approach (e.g., through Gillespie algorithms [20]) is known to be more realistic [18] and can significantly impact the simulated dynamics of infectious diseases (such as their seasonality [18]). Finally, an agent-based implementation is sometimes required to reach a level of detail that would not be tractable with other approaches because of combinatorial explosion [21].

One of the problems of modelling is bridging the gap between conceptual models (categories and their transition) and their computer simulation (through deterministic, stochastic or agent-based implementation). In fact, going from a conceptual model to a computational one requires some programming skills, which could be very rudimentary for the deterministic implementation to extremely complex for agent-based models. However, this gradient of programming complexity yields a high heterogeneity in the code that could be produced, with non negligible impacts on the disease dynamics. For instance, despite their simplicity, deterministic models can show very different dynamics according to the algorithm being used [18]. We also observe that the software implementation of such models and their accompanied modelling tools do not adhere to well established software engineering best practices such as *modularity* through *clear separation of concerns* that can promote *reproducibility* and *reusability* by other researchers.

Domain Specific Languages (DSLs) address such difficulties by separating modelling and specification concerns (conceptual model) from implementation aspects

(computational model). Contrary to General-purpose Programming Languages (GPLs), DSLs are higher-level languages that provide a more expressive syntax based on abstractions and notations that are directly-related to the concepts of the studied domain [22][23][24].

Our goal with **Kendrick** is to simplify the creation of epidemiological models through a unified Domain-Specific Language for epidemiology that can support a variety of modelling and simulation approaches used in the field. This goal can be achieved by promoting *reproducibility* and *reuse* with modular modelling abstractions.

Kendrick is designed according to a general meta-model that captures the basic domain concepts and opens up the possibility of taking into account different aspects of epidemiological modelling. To illustrate the practical usage of our modelling language, we present in this paper two case studies on measles and mosquito-borne diseases. Our approach is carefully validated through statistical comparisons between our simulation results and well-established platform simulations in order to guarantee the link with the mathematical epidemiology literature. While part of the mathematical and syntactical details of the **Kendrick model** and language have been presented before [25][26], this is the first time the software itself is presented, through a step by step coverage of its implementation and usage.

Implementation

Kendrick is implemented on top of the Pharo [27] programming environment using the Moose meta-modelling platform [28]. The numerical analysis back-end is based on the PolyMath framework [29], the visualisation sub-system on Roassal [30, 31] and the UI components on GTTools and the Moldable Inspector [32]. The DSL makes extensive use of Pharo's reflective sub-system [33], with on-going extensions based on the PetitParser [34] framework and Helvetia parsing workbench [35].

Kendrick supports all major desktop platforms (Gnu/Linux, Mac OS and MS Windows) and adopts an open continuous integration process (based on github[1], smalltalkhub[2] and Travis CI[3]). It is extensively tested using the SUnit [36] framework, readily available through a dedicated web-site [4] and well documented through a collaborative wiki [5].

This section focuses on the implementation details of the **Kendrick** meta-model and DSL. For step-by-step installation and experimentation details see the following Section (*i.e.* Results & Discussion).

The Kendrick Meta-model

As we saw, **Kendrick** aims to support a wide range of epidemiological models that may use either different formalisms or different implementation techniques.

Kendrick needs to abstract away from programming concerns and allow the definition of epidemiological models with little or no programming skills.

[1] `https://github.com/UMMISCO/kendrick`

[2] `http://smalltalkhub.com/#!/~UMMISCO/Kendrick`

[3] `https://travis-ci.org/UMMISCO/kendrick`

[4] `http://ummisco.github.io/kendrick/`

[5] `https://github.com/UMMISCO/kendrick/wiki`

In order to achieve these goals we had to identify the proper abstractions and implement them as a general core meta-model, which means implementing: *a set of domain concepts on which all possible base-level models and simulation tools could rely*. The resulting meta-model that we defined turned out to be general enough to capture compartment-based population dynamics rather than just epidemiology.

Seen from this perspective (*i.e.* of compartment-based population dynamics), models aim at answering the following question: *What is the cardinality of the defined compartments at any given time ? Or conversely: Given a set of observations, what parameters of an input model can best fit the observed behaviour ?*. To answer these questions the **Kendrick** meta-model (seen in Figure 1) includes the following concepts: *Population, Attribute, Equivalence Relation, Compartment, Parameter, Transition and Time Series*.

In order to be as general as possible, *compartments* are assumed to be defined by an *equivalence relationships* on the *population*. In the current version of the meta-model, this is done using the *attributes* of the individuals (such as the infectious status, species, age *etc.*). Moreover, in the implementation, the equivalent individuals are those with the same values for a given set of attributes.

The *Time series* concept captures the simulation results of a model over a certain period of time. Alternatively, partial times series can be used to capture observations. Finally, the defined models may also include additional *parameters* such as: rates, probabilities or initial cardinalities of the population or of compartments. These parameters are captured as numerical temporal functions.

It should be noted here that the **Kendrick** meta-model can capture both deterministic models expressed as a set of ODEs and demographic-stochastic models. Both kinds of models are stored as the transition rate matrix of a continuous-time Markov chain. Different formalisms can be used to write these *transitions*: such as ODEs or stochastic automata (with each syntax being able to be translated into the other). Finally, transition rates are depicted as temporal functions in our meta-model, rather than numbers, to capture variations of these rates (e.g. for seasonal forcing). Any given model can be run as a deterministic, stochastic or agent-based simulation. The default deterministic solver is RK4 [19]. Gillespie's direct and tau-leap methods are both available for stochastic simulations. Stochastic agent-based simulations can also be run by triggering events at the level of individuals [21].

The Kendrick DSL

As we saw, the **Kendrick** meta-model tries to address reusability in epidemiological modelling by decomposing highly-coupled monolithic epidemiological models into modular concerns. Indeed, separating these concerns is expected to make them easier to define, understand and change. It is achieved at a mathematical level by defining each concern as a stochastic automaton and by combining them using a tensor sum operator [37]. It is however still necessary to shield epidemiologist from implementation details and avoid to introduce dependencies among concerns in the implementation of models. The goal of the DSL is thus threefold: a) provide an easy-to-use, concise and readable syntax b) hide implementation details as much as possible c) provide a way to keep the concerns separated (i.e. avoid linguistic dependencies between them).

With these goals in mind, our DSL defines the following linguistic (syntactical and semantical) entities: Epidemiological *Models*, *Compositions*, *Scenarios*, *Simulations* and *Visualizations*, which are exemplified in Table 1. As we will see in greater detail in the following section, each one of these entities can be defined as a separate component (in its own file if necessary) and reused several times in multiple projects by simply referring to its name. Moreover, all of the aforementioned entities can be extended (*i.e.* specialized) upon reuse, by simply overriding one or more of their properties. *Models* are the more versatile and reusable entities, since as we can see in the Table below, they define general epidemiology concerns (such as the SIR dynamics or other general aspects, like the type and number of species being modeled).

These independent models can then be combined (through the *Composition* entity) by further refining cross-concern characteristics. *Scenarios* describe the initialization data of experiments, but can themselves be combined in different ways to provide input to *Simulation* entities. These latter entities define algorithmic properties for experiments, such as solvers and timing constraints. Finally, *Visualization* entities define the desired visual outputs (such as epidemiological figures and maps) that can summarize the experimental results.

Results & Discussion

In this section, we first provide practical information on how to install and start using **Kendrick**. Then we illustrate in detail how to simulate the evolution of two infectious diseases. The first one is measles, a childhood disease that has been extensively studied through epidemiological modelling. The second one is a vector-borne disease on which we test the multi-host component of our platform.

Installing & running Kendrick

The easiest way to install **Kendrick** is to download a pre-compiled bundle of the **0.42 version**, for your platform of choice (Mac, Linux and Windows are supported) by following the corresponding links in the Github repository [6]. After unzipping the corresponding .zip file for your platform, you can go ahead and double-click on one of the kendrick launchers (**KendrickUI** for Mac and Linux, or **Kendrick-WinLauncher** for Windows).

Alternatively, we provide a very straight-forward method to compile **Kendrick** from sources on all systems with a bash command-line (this includes Linux, Mac and Windows with Cygwin and/or the Windows 10 Bash sub-system), by issuing the following command:

```
wget -O- https://goo.gl/WUQxmp | bash
```

This command will automatically retrieve all the required dependencies, compile **Kendrick** from sources and set up all execution scripts for normal or development use. After compiling or downloading the pre-compiled versions of **Kendrick**, the dedicated editor can be run by invoking:

```
./KendrickUI
```

[6] `https://github.com/UMMISCO/kendrick`

In which case you will be greeted with the splash screen (Figure 2).

Our dedicated editor is a moldable editor / inspector [32] that follows the Cascading Lists [7] pattern of navigation. This means that each selection reveals a brand new column of actionable information (on the right), with a horizontal bar on the bottom controlling the viewport position and size.

Alternatively, to run **Kendrick** with a full development environment (allowing to use both the DSL and the Pharo API of **Kendrick**), you can invoke:

```
./KendrickDevUI
```

Finally, to use **Kendrick** with an editor of your choice, you only need to navigate in the Sources directory of your installation, edit or add files for your project and invoke the non-interactive **Kendrick** executable as follows (example for simulating and visualizing the results described in Influenza1Viz.kendrick on your sources folder):

```
./Kendrick Sources/Projects/Influenza/Visualization/Influenza1Viz.kendrick
```

In this case final visualization results will be produced in:

```
Sources/Projects/Influenza/Output/Influenza1Viz.png
```

Case-study I: Measles

We consider the transmission of measles through a SEIR model with demography [38]. Individuals are born with the *Susceptible (S)* status with a birth rate μ, then may become *Exposed (E)* (i.e. infected but not yet infectious) with transmission rate βI. After an average latent period, $1/\sigma$, they may become *Infectious (I)* and may finally switch to *Recovered (R)* after an infectious period: $1/\gamma$.

The model is available through the standard **Kendrick** distribution by navigating through our editor either to Scripts/Measles.kendrick (single file script, seen in Figure 3) or to Projects/Measles (several decomposed entities in separate files that are more easily extended and reused).

The scripting version of the model is shown below:

```
KendrickModel SIR
  attribute: #(status -> S I R);
  parameters: #(beta lambda gamma mu);
  transitions: #(
    S -- lambda --> I.
    I -- gamma --> R.
    status -- mu --> Empty.
    Empty -- mu --> S.
  ).
```

[7] http://designinginterfaces.com/firstedition/index.php?page=Cascading_Lists

```
KendrickModel SEIR                                                    11
  extends: 'SIR';                                                     12
  parameters: #(sigma);                                               13
  delay: #(sigma , S -- lambda --> I , E).                            14
                                                                      15
Composition Measles                                                   16
    model: 'SEIR'.                                                    17
                                                                      18
Scenario MeaslesParameters                                            19
    on: 'Measles';                                                    20
    beta: 0.0000214;                                                  21
    gamma: 0.143;                                                     22
    mu: 0.0000351;                                                    23
    sigma: 0.125;                                                     24
    lambda: #(beta*I).                                                25
                                                                      26
Scenario MeaslesPopulation                                            27
    on: 'Measles';                                                    28
    populationSize: 100000;                                           29
    S: 99999;                                                         30
    I: 1;                                                             31
    others: 0.                                                        32
```

Where we can see lines 1 to 5 defining the SIR Epidemiological concern and the named transition parameters between its compartments. Then from lines 11 to 14 the SIR concern is extented to produce the SEIR compartmental description by introducing a delay parameter (sigma) and a compartment (E) between S and I. Subsequently, in lines 16 to 19 the Measles model is produced by using the SEIR concern while defining the formula for computing lambda. Complete models (such as Measles) are produced by the Composition entity of the DSL which as we saw on Table 1 can combine several different modelling concerns. Finally, from lines 21 to 33 two experiment scenarios are declared (one describing parameter values and the other population data) each of which can be used independently and combined with other initialization descriptions.

This version of the Measles model relies on the transition syntax of **Kendrick**, but alternative formalisms such as ODEs are also possible as seen below:

```
KendrickModel SEIR                                                    1
  attribute: #(status -> S E I R);                                    2
  parameters: #(beta sigma gamma mu);                                 3
  equations: #(                                                       4
      S:t=mu*N - beta*S*I - mu*S.                                     5
      E:t=beta*S*I - sigma*E - mu*E.                                  6
      I:t=sigma*E - gamma*I - mu*I.                                   7
      R:t=gamma*I - mu*R.                                             8
  ).                                                                  9
```

We should note here that frequently used entities such as SIR or SEIR shown above are part of the standard **Kendrick** library (available through our editor by navigating to Library/KendrickModel, seen in Figure 4).

The initial size of the population is 100,000 individuals and the parameters above, are taken directly from the related literature [18, 38]. Figure 5 shows the result of the deterministic simulation (using the RK4 solver) produced by the Simulation and Visualization entities shown below:

```
Simulation MeaslesRKSim rungeKutta                                    1
  scenarios: #(MeaslesParameters MeaslesPopulation);                  2
  from: 0.0;                                                          3
  to: 150;                                                            4
  step: 1.                                                            5
                                                                      6
Visualization MeaslesDiagramViz diagram                               7
  for: 'MeaslesRKSim';                                                8
  xLabel: 'Time (days)';                                              9
  exportToPng.                                                        10
```

The Simulation entity (from lines 1 to 5) uses the two scenarios we defined previously, in addition of defining the specific algorithm to be used (rungeKutta) and the timing and stepping charecteristics. Finally, the Visualization entity specifies the desired output for a specific Simulation and by default will plot the dynamics of the Infected compartment over time.

Case-study II: Mosquito-borne disease with three host species

As a second example we will study an SIR model with demography of a mosquito-borne disease with three host species. These species include a vector species (mosquito) and two potential host species species, designated as reservoir1 and reservoir2. The population is partitioned using two attributes: *status* and *species* leading to an arrangement of 3x3 compartments.

All species have the same six transitions: birth, deaths (for each one of the three status compartments), infection and recovery. Given a transition, the probability function is the same for everyone. Only 4 generic transitions need to be defined with **Kendrick**, which will then automatically generate the remaining transitions for each species of the model.

As previously the model is available through the standard **Kendrick** distribution by navigating through our editor either to Scripts/Mosquitos.kendrick (single file script, seen in Figure 6) or to Projects/Mosquito.

The scripting version of the model is shown below.

```
KendrickModel SIR                                                     1
  attribute: #(status -> S I R);                                      2
  parameters: #(beta lambda gamma mu);                                3
  transitions: #(                                                     4
    S -- lambda --> I.                                                5
```

```
    I -- gamma --> R.
    status -- mu --> Empty.
    Empty -- mu --> S.
  ).

KendrickModel Species
  parameters: #(rho);
  attribute: #(species -> mosquito reservoir1 reservoir2).

Composition Mosquito
  model: 'SIR';
  model: 'Species'.

Scenario MosquitoPopulation
  on: 'Mosquito';
  populationSize: 13000;
  S_species: #(9999 1000 2000);
  I_species: #(1 0 0).

Scenario MosquitoParameters
  on: 'Mosquito';
  gamma: 52;
  beta: 1;
  mu_species: #(12.17 0.05 0.05);
  rho_species: #(
    #(0 0.02 0.02)
    #(0.02 0 0)
    #(0.02 0 0)
  );
  N: #(species);
  lambda: #(beta*rho*I sum).

Simulation MosquitoGillepsie gillespie
    scenarios: #(MosquitoPopulation MosquitoParameters);
    from: 0.0;
    to: 0.5;
    step: 0.0027.

Visualization MosquitoDiagramViz diagram
  for: 'MosquitoGillepsie';
  data: #(I sqrt);
  xLabel: 'Time (days)';
  exportToPng.
```

Lines 1 to 9 define the SIR example with demography (also available as a library model). Then on lines 11 to 13 the ecological concern is described, with the 3

species mentioned above, including the related rho parameter of our model. The composition of the epidemiological and ecological concerns takes place on lines 15 to 20, where the population N and the lambda parameters are declared. Two scenario entities follow (on lines 22 to 37) covering the population initialization for the different compartments, as well as the concrete parameter values for the rest of the model. The Simulation entity uses the two scenarios defined above to initialize the Gillespie algorithm for the given time-frame and stepping characteristics. Finally, the Visualization entity specifies the infectious stochastic dynamics to be plotted, over time (as seen on Figure 7).

Validating the implementation of our language and simulation platform

To validate the core functionalities of **Kendrick** as a modelling and simulation platform, we first compared the output of deterministic **Kendrick** models with reference implementations of the same models on Scilab [39]. The simulation results (as those shown in Figure 5) suggest that for deterministic models **Kendrick** produces identical results to the reference implementations.

The dynamics of deterministic simulations have also been compared with those of Gillespie-based and agent-based simulations (Figure 8).

The results for the measles model (taking into account scenarios with and without vaccination) can be seen in the upper part of Figure 8, with the lower part displaying the results of the mosquito-borne model for each individual host species. The deterministic dynamics can be superimposed on the stochastic and agent-based ones, validating our implementation of the stochastic and agent-based simulation logic.

Finally, we cross-examined agent-based and stochastic simulation outputs with the same configuration. Given that each agent-based and stochastic model is executed 200 times, we extracted from the simulation results key properties of the epidemiological dynamics, such as: epidemic peak, time at epidemic peak, and epidemic duration. A Kolmogorov-Smirnov statistical test on each pair of samples confirmed that there is no statistical difference between the agent-based simulations and the stochastic ones. These test results can be seen in Table 2, where we observe all $P - values$ being greater than 0.05, validating that the resulting distributions are statistically indistinguishable. This in turn shows us the expected equivalence of the two algorithms, re-affirming the correctness of their implementation.

Discussion

Although DSLs have been used before in the context of bioinformatics [40, 41, 42], only a small number of them focused on epidemiological modelling [43, 44]. For example, *Ronald* [43] is a DSL for studying the interactions between malaria infections and drug treatments, but has unfortunately been discontinued. Schneider and al. [44] also proposed a DSL for epidemics, but their solution only supported agent-based models. Mathematical modelling languages (MMLs) such as Scilab [39], Modelica [45], Matlab [46] or JSim [47] do allow easier definition of mathematical models as sets of ODEs, but are too broad in scope to properly cover the domain-specific needs of epidemiology. On the other hand, individual libraries targeting epidemiological modelling: such as Epipy - a visualisation data tool for epidemiology

written in Python [48], or GillespieSSA - an R package for generating stochastic simulation using Gillespie's algorithms [49], cover only very specific sub-problems and epidemiological needs.

Closer to our approach are computational modelling tools for epidemiology such as FluTe [50], GLEAMviz [51], STEM [52] and FRED [53]. These solutions use dedicated approaches to model the transmission of infectious disease and provide a graphical user interface (GUI) to specify and visualise an epidemiological model. The main features of these tools are summarised in Table 3 and compared to **Kendrick**.

Contrary to solutions that only focus on stochastic simulations (such as GLEAMviz [51] and STEM [52]) our platform can provide more detailed modelling thanks to the availability of agent-based simulation. Investigation of epidemics at a global scale as provided by GLEAMviz [51], STEM [52] or FRED [53] is also possible in **Kendrick**. A case study of meta-population model can be found in the additional file S1 provided with this publication. This particular case is also an example of spatial visualisation integrated within the platform.

Given the above comparison, **Kendrick** can be considered as a higher-level disciplined solution that aims to cover as many specificities of epidemiological modelling as possible. The fact that **Kendrick** chooses the right level of abstraction for each case, also helps modellers to focus on what is essential and avoid irrelevant or inconsistent definitions. This allows for easier cross-examination of different modelling paradigms (such as deterministic, demographic-stochastic and agent-based). From the modellers point of view, working directly with epidemiological concepts such as compartments, is more familiar than the underlying mathematical abstractions and algorithmic implementations.

In order to provide these abstractions in a transparent way for the user, our core model divides the population of **Kendrick** models in homogeneous classes, using mathematical equivalence relations (described in more details in [54]). By doing so we are able to capture different scenarios of infectious diseases such as multi-hosts, multi-strains[55, 21], heterogeneity, meta-population[2, 10] or network of contacts[1].

Conclusion

In this paper, we have introduced a modelling language and simulation platform that allows specifying and simulating epidemiological models. The goal of our modelling approach is to allow experts to focus on their conceptual vocabulary through a dedicated DSL for epidemiology. We have highlighted the capacities of our platform through two classic examples of epidemiological models and we showed that the dynamics produced by our software are equivalent to well-established but harder to use, programming platforms.

By generalising the domain concepts used by experts, we have constructed a general language meta-model for epidemiology. This allowed us to express various epidemiological concerns such as spatial transmission, mobility, heterogeneous populations, evaluation of control strategies, visualisation, etc... in a uniform manner. We thus hope that our platform will be further adopted and extended to consider even more facets of epidemiology.

The **Kendrick** platform is available as an open source software under the MIT licence: `http://ummisco.github.io/kendrick/`.

1 Declarations

Availability of data and materials
Project name: KENDRICK
Project home page: `http://ummisco.github.io/kendrick/`
Operating System: multi-platform (Linux/Mac OS X/Windows)
Programming environment: Pharo 6.1: `http://www.pharo.org/`
Requirements: All the required tools for the installation of KENDRICK are described on the project home page.
License: MIT License

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
BTMA participated in study conception, carried out the implementation of the software, carried out the experimentation of the models, analysis of simulation results and drafted the manuscript. NP participated in study conception, carried out the DSL implementation and improved the manuscript. BR participated in study conception, to carry out the experimentation of models, results analysis and improved the manuscript. SS participated in study conception, carried out the implementation of the software and improved the manuscript. MZ participated in study conception and improved the manuscript. All authors read and approved the final manuscript.

Acknowledgements
This work was supported by Alexandre Bergel (Chile University) and his team Object Profile (`http://objectprofile.com/ObjectProfile.html`) with the visualisation tool ROASSAL. We would like to acknowledge the financial support from Agence Nationale de la Recherche (ANR) PANIC project and European Smalltalk User Group (ESUG). We also gratefully acknowledge the financial support from Hanoi University of Science and Technology through the "Spatial Modelling Language for Epidemiology" project - grant number T2017-TT-001.

Author details

[1]Software Engineering Department, School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam. [2]Université de La Rochelle, UMR 7266 LIENSs, CNRS, La Rochelle, France. [3]Sorbonne Université, IRD, Unité de Modélisation Mathématiques et Informatique des Systèmes Complexes, UMMISCO, F-93143 Bondy, France. [4]Université de Yaoundé I, IRD, UMMISCO, Yaoundé, Cameroon. [5]Université de Caen Basse-Normandie, Caen, France. [6]Université Sorbonne Paris Cité, Université Paris Descartes, Paris, France. [7]Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005, Paris, France.

References

1. Keeling, M.J., Eames, K.T.D.: Networks and epidemic models. J R Soc Interface **2(4)**, 295–307 (2005)
2. Xia, Y., Bjornstad, O.N., Grenfell, B.T.: Measles metapopulation dynamics: a gravity model for epidemiological coupling and dynamics. Am Nat **164(2)**, 267–281 (2004)
3. Gandon, S., Mackinnon, M.J., Nee, S., Read, a.F.: Imperfect vaccines and the evolution of pathogen virulence. Nature **414(6865)**, 751–756 (2001)
4. Read, A.F., Huijben, S.: Perspective: Evolutionary biology and the avoidance of antimicrobial resistance. Evolutionary Applications **2(1)**, 40–51 (2009)
5. Bauch, C.T., Szusz, E., Garrison, L.P.: Scheduling of measles vaccination in low-income countries: Projections of a dynamic model. Vaccine **27(31)**, 4090–4098 (2009)
6. Levin, A., Burgess, C., Garrison, L.P., Bauch, C., Babigumira, J., Simons, E., Dabbagh, A.: Global eradication of measles: an epidemiologic and economic evaluation. The Journal of infectious diseases **204 Suppl (Suppl 1)**, 98–106 (2011)
7. Jones, K.E., Patel, N.G., Levy, M.A., Storeygard, A., Balk, D., Gittleman, J.L., Daszak, P.: Global trends in emerging infectious diseases. Nature **451**, 990–994 (2008)
8. Morens, D.M., Fauci, A.S.: Emerging infectious diseases: Threats to human health and global stability. PLoS Pathog **9(7)** (2013)
9. Ferguson, N.M., Keeling, M.J., Edmunds, W.J., Gani, R., Grenfell, B.T., Anderson, R.M., Leach, S.: Planning for smallpox outbreaks. Nature **425(6959)**, 681–685 (2003)
10. Ferguson, N.M., Cummings, D.A.T., Cauchemez, S., Fraser, C., Riley, S., Meeyai, A., ... Burke, D.S.: Strategies for containing an emerging influenza pandemic in southeast asia. Nature **437(7056)**, 209–214 (2005)
11. Agur, Z., Cojocaru, L., Mazor, G., Anderson, R.M., Danon, Y.L.: Pulse mass measles vaccination across age cohorts. In: Proceedings of the National Academy of Sciences of the United States of America, vol. 90 (24), pp. 11698–11702 (1993)
12. Cohen, M.L.: Changing patterns of infectious disease. Nature **406**, 762–768 (2000)
13. Eisenberg, J.N.S., Cevallos, W., Ponce, K., Levy, K., Bates, S.J., Scott, J.C., ... W, L.: Environmental change and infectious disease: How new roads affect the transmission of diarrheal pathogens in rural ecuador. In: Proc Natl Acad Sci U S A (2006)
14. Ostfeld, R.S., Keesing, F.: Effects of host diversity on infectious disease. Annual Review of Ecology, Evolution, and Systematics **43(1)**, 157–182 (2012)
15. Keesing, F., al.: Impacts of biodiversity on the emergence and transmission of infectious diseases. Nature **468**, 647–652 (2010)
16. Laxminarayan, R., al.: Antibiotic resistance - the need for global solutions. The Lancet infectious diseases **13(12)**, 1057–1098 (2013)

17. Anderson, R.M., May, R.M.: Infectious Diseases of Humans: Dynamics and Control. Oxford Science Publications, Oxford (1991)
18. Keeling, M.J., Rohani, P.: Modeling Infectious Diseases. Princeton University Press, Princeton (2008)
19. Griffiths, D.F., Higham, D.J.: Numerical Methods for Ordinary Differential Equations. Springer, Springer Undergraduate Mathematics Series (2010)
20. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry **81**, 2340–2361 (1977)
21. Roche, B., Drake, J.M., Rohani, P.: An agent-based model to study the epidemiological and evolutionary dynamics of influenza viruses. BMC Bioinformatics **12**, 87 (2011)
22. Van Deursen, A., Klint, P., Viser, J.: Domain-specific languages: An annotated bibliography. ACM SIGPLAN Notices **35 (6)**, 26–36 (2000)
23. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys **37 (4)**, 316–344 (2005)
24. Fowler, M.: Domain-specific Languages. Pearson Education, USA (2010)
25. BUI, T.M.A., Stinckwich, S., Ziane, M., Roche, B., HO, T.V.: Kendrick: a domain specific language and platform for epidemiological modelling. In: 11th IEEE-RIVF International Conference on Computing and Communication Technologies. RIVF-2015, Cantho, Vietnam (2015)
26. Bui, T.M.A., Papoulias, N., Ziane, M., Stinckwich, S.: Explicit composition constructs in dsls: The case of the epidemiological language kendrick. In: Proceedings of the 11th Edition of the International Workshop on Smalltalk Technologies. IWST'16, pp. 20–12011. ACM, New York, NY, USA (2016)
27. Black, A.P., Ducasse, S., Nierstrasz, O., Pollet, D., Cassou, D., Denker, M.: Pharo by Example, p. 333. Square Bracket Associates, Kehrsatz, Switzerland (2009). http://pharobyexample.org/
28. Girba, T.: The Moose Book. Self Published, Switzerland (2010). http://www.themoosebook.org/book
29. Besset, D.H.: Object-oriented Implementation of Numerical Methods: an Introduction with Java and Smalltalk. Morgan Kaufmann, USA (2001)
30. Araya, V.P., Bergel, A., Cassou, D., Ducasse, S., Laval, J.: Agile visualization with roassal. Deep Into Pharo, 209–239 (2013)
31. Bergel, A.: Agile Visualization. Lulu, Chile (2016)
32. Chiș, A.: Moldable tools. PhD thesis, PhD thesis. University of Bern (2016)
33. Foote, B., Johnson, R.E.: Reflective facilities in smalltalk-80. In: ACM Sigplan Notices, vol. 24, pp. 327–335 (1989). ACM
34. Renggli, L., Ducasse, S., Gîrba, T., Nierstrasz, O.: Practical dynamic grammars for dynamic languages. In: 4th Workshop on Dynamic Languages and Applications (DYLA 2010) (2010)
35. Renggli, L., Gîrba, T., Nierstrasz, O.: Embedding languages without breaking tools. In: European Conference on Object-Oriented Programming, pp. 380–404 (2010). Springer
36. Ducasse, S.: SUnit Explained. `http://www.iam.unibe.ch/\texttildelowducasse/Programmez/OnTheWeb/SUnitEnglish2.pdf`
37. Bui, T.M.A., Ziane, M., Stinckwich, S., Ho, T.V., Roche, B., Papoulias, N.: Separation of concerns in epidemiological modelling. In: Companion Proceedings of the 15th International Conference on Modularity. MODULARITY Companion 2016, pp. 196–200. ACM, New York, NY, USA (2016)
38. Anderson, R.M., May, R.M.: Infectious Diseases of Humans vol. 1. Oxford university press Oxford, UK (1991)
39. Scilab, Open Source Software for Numerical Computation. `http://www.scilab.org/`
40. Fall, A., Fall, J.: A domain-specific language for models of landscape dynamics. Ecological Modelling **141**, 1–18 (2001)
41. Degenne, P., Lo Seen, D., Parigot, D., Forax, R., Tran, A., Ait Lahcen, A., Curé, O., Jeansoulin, R.: Design of a domain specific language for modelling processes in landscapes. Ecological Modelling **220**, 3527–3535 (2009)
42. van Engelen, R.A.: Atmol: A domain-specific language for atmospheric modelling. CIT Journal of Computing and Information Technology **9**, 289–303 (2001). Special Issue on Domain-Specific Languages Part I
43. Antao, T., Hastings, I., McBurney, P.: Ronald: A domain-specific language to study the interactions between malaria infections and drug treatments. In: Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, pp. 785–790 (2012)
44. Schneider, O., Dutchyn, C., Osgood, N.: Towards frabjous: a two-level system for functional reactive agent-based epidemic simulation. In: Proceedings of the 2008 International Conference on Bioinformatics and Computational Biology, BIOCOMP, pp. 747–752 (2008)
45. Modelica Language. `https://www.modelica.org/`
46. Matlab, the Language of Technical Computing. `http://www.mathworks.com/products/matlab/`
47. Introduction of JSim Framework. `http://www.physiome.org/jsim/`
48. Epipy, Python Tools for Epidemiology. `http://cmrivers.github.io/epipy/`
49. GillespieSSA: Gillespie's Stochastic Simulation Algorithm (SSA). `http://cran.r-project.org/web/packages/GillespieSSA/index.html`
50. Chao, D.L., Halloran, M.E., Obenchain, V.J., Longini, I.M.J.: Flute, a publicly available stochastic influenza epidemic simulation model. PLoS Comput Biol **6** (2010)
51. Van den Broeck, W., Gioannini, C., Gonçalves, B., Quaggiotto, M., Colizza, V., Vespignani, A.: The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. BMC Infectious Diseases **11**, 37 (2011)
52. Falenski, A., Filter, M., Thöns, C., Weiser, A.A., Wigger, J.-F., Davis, M., Douglas, J.V., Edlund, S., Hu, K., Kaufman, J.H., *et al.*: A generic open-source software framework supporting scenario simulations in bioterrorist crises. Biosecurity and bioterrorism: biodefense strategy, practice, and science **11**(S1), 134–145 (2013)
53. Grefenstette, J.J., Brown, S.T., Rosenfeld, R., DePasse, J., Stone, N.T.B., Cooley, P.C., Wheaton, W.D., Fyshe, A., Galloway, D.D., Sriram, A., *et al.*: Fred (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. BMC public health **13**(1), 940 (2013)

54. Bui, T.-M.-A.: Separation of concerns in Epidemiology. Theses, Université Pierre et Marie Curie - Paris VI (December 2016). `https://tel.archives-ouvertes.fr/tel-01506726`
55. Reiner, R.C., Perkins, T.A., Barker, C.M., Niu, T., Chaves, L.F., Ellis, A.M., George, D.B., Le Menach, A., Pulliam, J.R., Bisanzio, D., *et al.*: A systematic review of mathematical models of mosquito-borne pathogen transmission: 1970-2010. Journal of The Royal Society Interface **10**(81), 20120921 (2013)

Figures

Figure 1 The meta-model UML diagram This UML diagram shows main elements of our core systems and the relationships among them. Each element in this diagram represents a concept of our language. The composition link between two concepts indicates that the construction of a concept (where the filled diamond of the link points) may involve other ones (at other side of this link).

Figure 2 Kendrick UI: Running the Kendrick DSL Editor

Figure 3 Kendrick UI: Running the Measles Script

Figure 4 Kendrick UI: The Entity Library

Figure 5 Deterministic simulation of the measles model $S = 99999$, $E = 0$, $I = 1$, $R = 0$, $\beta = 0.0000214$, $1/\gamma = 7$ days, $1/\sigma = 8$ days, $\mu = 1/(78 * 365)$ in day^{-1}, $N = 100000$. The graph shows the infectious deterministic dynamics of the measles model using the KENDRICK language.

Figure 6 Kendrick UI: Running the Mosquitos Script

Figure 7 Stochastic simulation of the mosquito-borne disease The multi-host model with three species: mosquito, reservoir1, reservoir2. $S_1 = 9999$ (mosquito), $S_2 = 1000$ (reservoir 1), $S_3 = 2000$ (reservoir 2); $I_1 = 1$, $I_{2,3} = 0$, $R_{1,2,3} = 0$, $N_1 = 10000$, $N_2 = 1000$, $N_3 = 2000$; $\sigma = 52$, $\mu_1 = 365/30$, $\mu_{2,3} = 1/20$, $\beta_{12} = \beta_{13} = 0.02$, $\beta_{others} = 0.0$.

Figure 8 Comparison between the dynamics of deterministic, stochastic and agent-based model We show the simulation results of two models in three formalisms: deterministic (green line), stochastic (blue lines) and agent-based (red lines). The first row shows the results of the measles model. The measles model is on the left hand side and the vaccination model is on the right hand side. The second row shows the results of the mosquito-borne model with three host species

Tables

Additional Files

Additional file 1 (s1) — The meta-population model with immigration infection specified in KENDRICK language

Table 1 Kendrick DSL Entities (Two species Influenza SIR Example)

Entities	Syntax Examples

Model	`KendrickModel SIR`
	`  attribute: #(status -> S I R);`
	`  parameters: #(beta lambda gamma mu);`
	`  transitions: #(`
	`  S -- lambda --> I.`
	`  I -- gamma --> R.`
	`  status -- mu --> Empty.`
	`  Empty -- mu --> S.).`
	
	`KendrickModel TwoSpecies`
	`  attribute: #(species -> human bird).`

Composition	`Composition Influenza`
	`  model: 'TwoSpecies';`
	`  model: 'SIR'.`

Scenario	`Scenario InfluenzaPopulation`
	`  on: 'Influenza';`
	`  populationSize: 5500;`
	`  S_species: #(500 4990);`
	`  I_species: #(0 10).`
	
	`Scenario InfluenzaParameters`
	`  on: 'Influenza';`
	`  lambda: #(beta*I/N sum);`
	`  N: #(species);`
	`  mu_species: #(0.0000365 0.00137);`
	`  gamma_species: #(0.25 0.233);`
	`  beta_species: #(#(0 0.21) #(0 0.42)).`

Simulation	`Simulation InfluenzaRK rungeKutta`
	`  scenarios: #(`
	`  InfluenzaParameters`
	`  InfluenzaPopulation);`
	`  from: 0;`
	`  to: 500;`
	`  step: 1.`

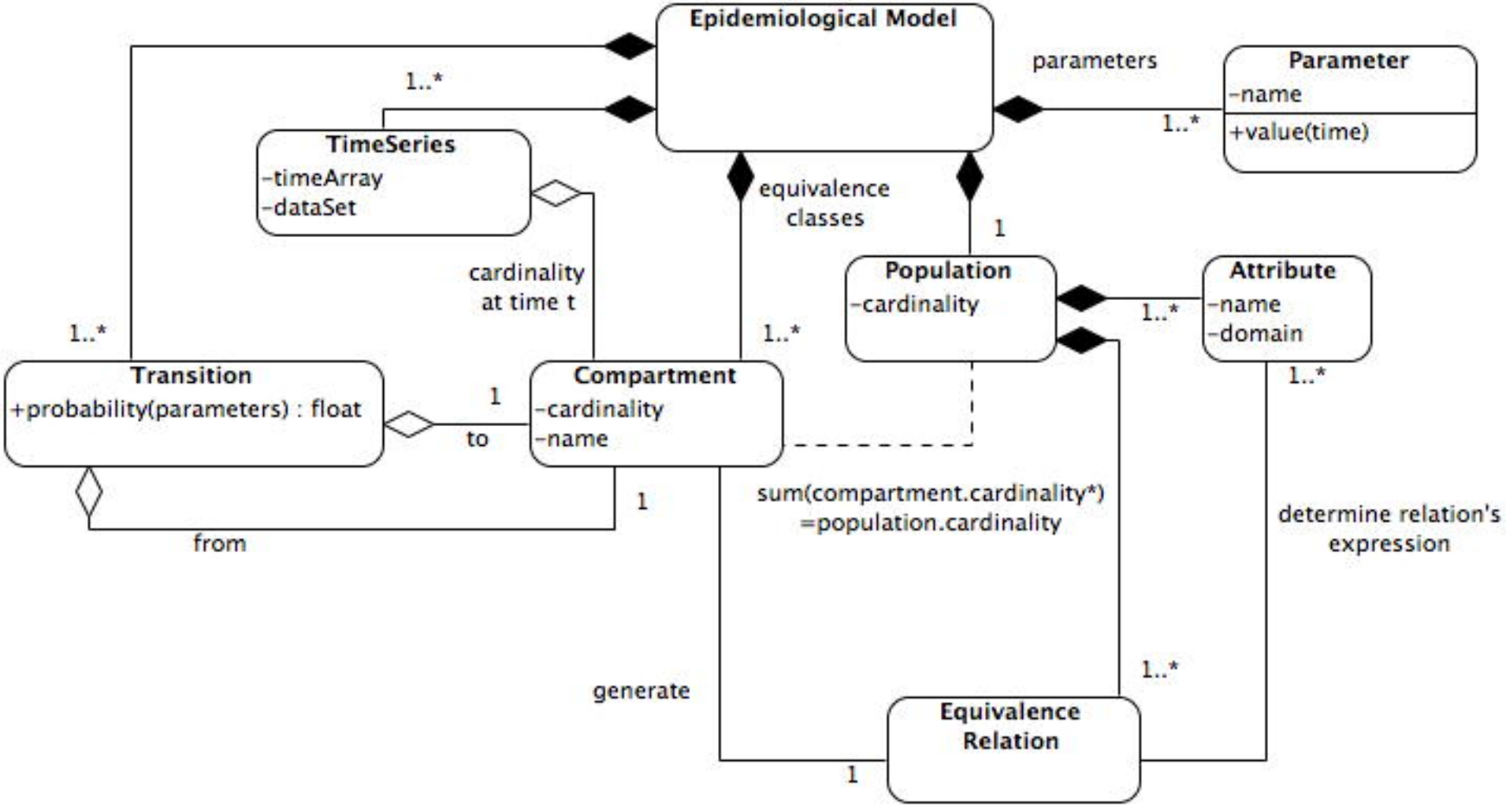
Visualization	`Visualization InfluenzaViz diagram`
	`  for: 'InfluenzaRK';`
	`  xLabel: 'Time (days)';`
	`  exportToPng.`

Table 2 P-values of Kolmogorov-Smirnov Test on two models over some disease global properties

Model		Global properties of diseases		
		Peak of epidemic	Time at peak of epidemic	Epidemic duration
Measles	Without Vaccination	0.8762	0.2471	0.5246
	With Vaccination	0.8928	0.2467	0.6262
Mosquito-borne disease	Species 1	0.7920	0.2700	0.7112
	Species 2	0.6272	0.3927	0.8643
	Species 3	0.7920	0.1122	0.2202

Table 3 Modelling and Simulation Tools for epidemiology

Criteria		KENDRICK	FluTe	GLEAMviz	STEM	FRED
Modelling approach	Deterministic	+	-	-	+	-
	Stochastic	+	-	+	+	-
	Agent-based	+	+	-	-	+
Population	Multi-species	+	-	-	+	-
	Heterogeneous	+	+	-	-	+
Spatial Structure	Patch model	+	+	+	+	-
	Network of contact	+	-	-	-	+
	Mobility pattern	+	+	+	+	+
Intervention Strategies		+	-	+	+	+
Simulation at global scale		-	-	+	+	+


Epidemiological Model
parameters
Parameter
-name
+value(time)
1..*
1..*
TimeSeries
-timeArray
-dataSet
equivalence
classes
1
cardinality
at time t
Population
-cardinality
Attribute
-name
-domain
1..*
1..*
1..*
1..*
Transition
+probability(parameters) : float
1
to
Compartment
-cardinality
-name
1
from
sum(compartment.cardinality*)
=population.cardinality
determine relation's
expression
generate
1..*
Equivalence
Relation
1

kendrick.image

Sources/

Folder

Entity

- Library
- Projects
- Scripts

Information
EvaluateCommandLineHandler successfully finished

Scripts

Folder

Entity

- Output
- Ebola.kendrick
- Influenza1.kendrick
- Influenza2.kendrick
- Influenza3.kendrick
- Influenza4.kendrick
- Influenza5.kendrick
- Influenza6.kendrick
- Measles.kendrick
- Mosquitos.kendrick

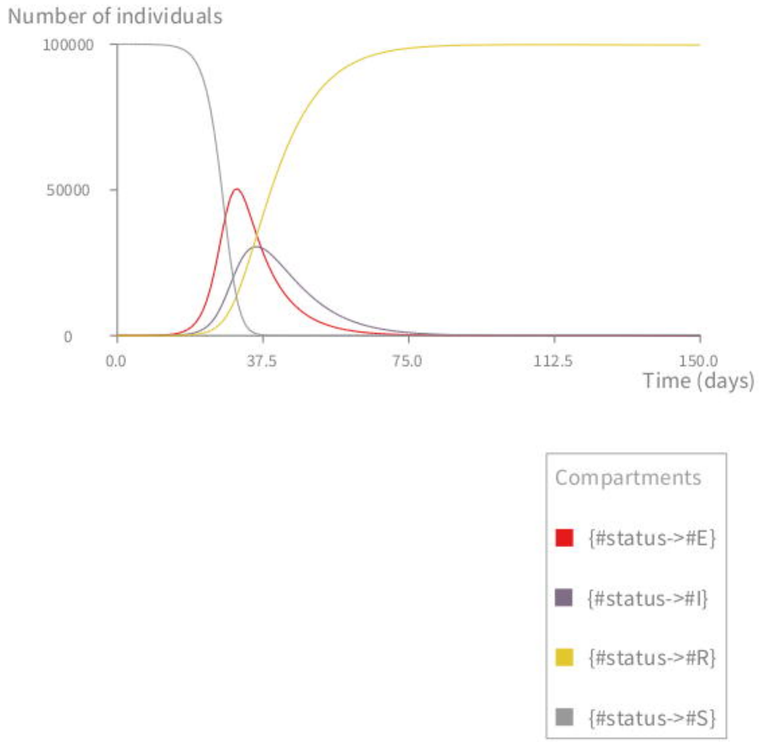
Measles.kendrick

Code

```
KendrickModel SIR
   attribute: #(status -> S I R);
   parameters: #(beta lambda gamma mu);
   transitions: #(
      S -- lambda --> I.
      I -- gamma --> R.
      status -- mu --> Empty.
      Empty -- mu --> S.
   ).

KendrickModel SEIR
   extends: 'SIR';
   parameters: #(sigma);
   delay: #(sigma , S -- lambda --> I  , E).

Composition Measles
    model: 'SEIR'.

Scenario MeaslesParameters
    on: 'Measles';
    beta: 0.0000214;
    gamma: 0.143;
    mu: 0.0000351;
    sigma: 0.125;
    lambda: #(beta*I).


Scenario MeaslesPopulation
    on: 'Measles';
    populationSize: 100000;
    S: 99999;
    I: 1;
    others: 0.

Simulation MeaslesRKSim rungeKutta
   scenarios: #(MeaslesParameters MeaslesPopulation);
   from: 0.0;
   to: 150;
   step: 1.

Visualization MeaslesDiagramViz diagram
   for: 'MeaslesRKSim';
   xLabel: 'Time (days)';
   exportToPng.
```

TRCanvas

Canvas



KendrickModel

Folder

Entity

- SEIIRS.kendrick
- SEIR.kendrick
- SEIRS.kendrick
- SEIRSQ.kendrick
- SIR.kendrick

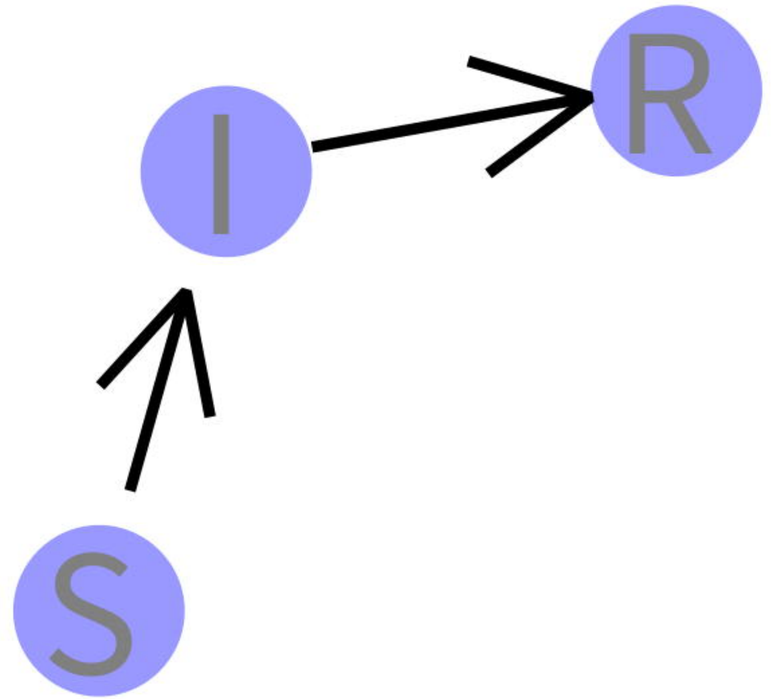
SIR.kendrick

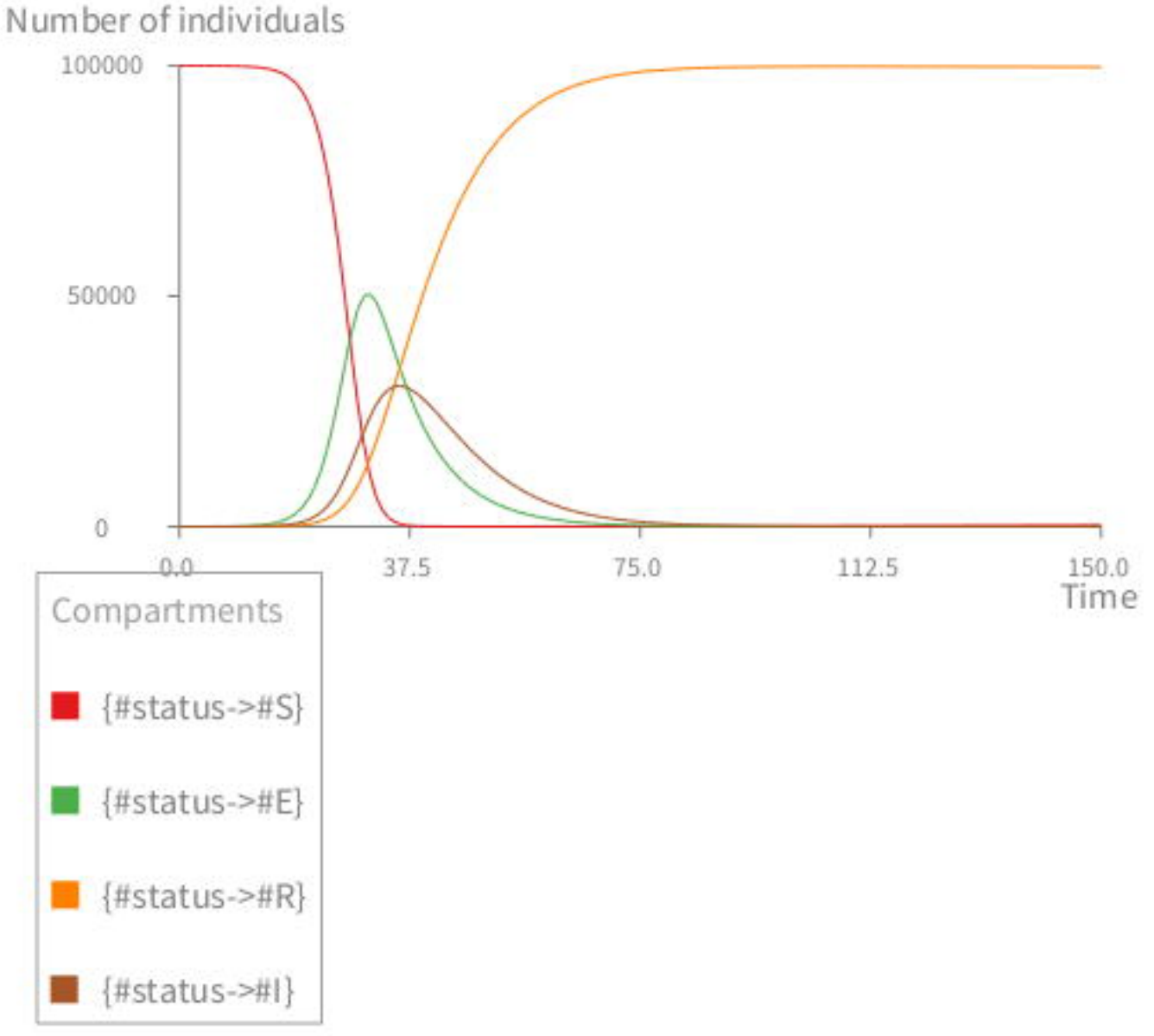
Code

```
KendrickModel SIR
   attribute: #(status -> S I R);
   parameters: #(beta lambda gamma mu);
   transitions: #(
      S -- lambda --> I.
      I -- gamma --> R.
      status -- mu --> Empty.
      Empty -- mu --> S.
   ).
```

KendrickModel

Graph




Number of individuals
100000
50000
0
0.0
37.5
75.0
112.5
150.0
Time
Compartments
{#status->#S}
{#status->#E}
{#status->#R}
{#status->#I}

Scripts

Folder

Entity

- Output
- Ebola.kendrick
- Influenza1.kendrick
- Influenza2.kendrick
- Influenza3.kendrick
- Influenza4.kendrick
- Influenza5.kendrick
- Influenza6.kendrick
- Measles.kendrick
- Mosquitos.kendrick

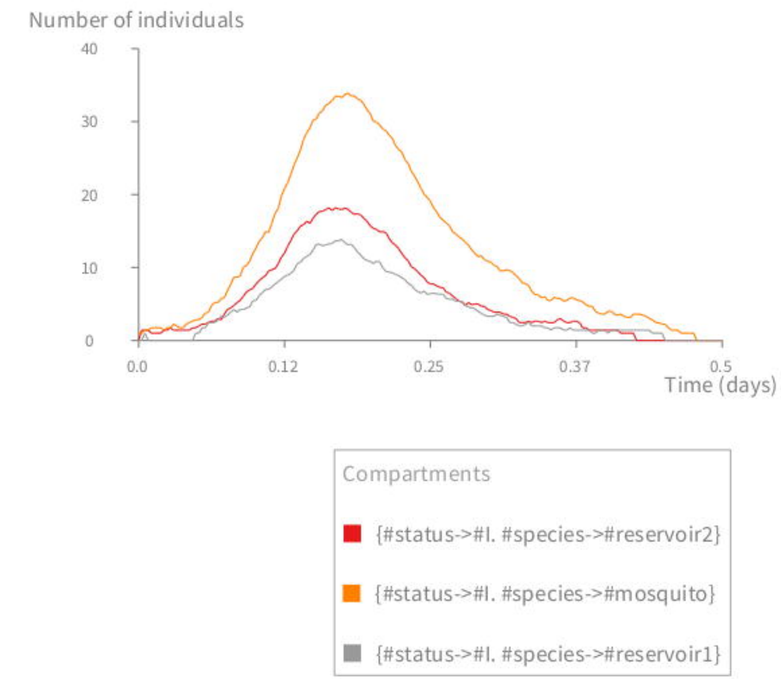
Mosquitos.kendrick

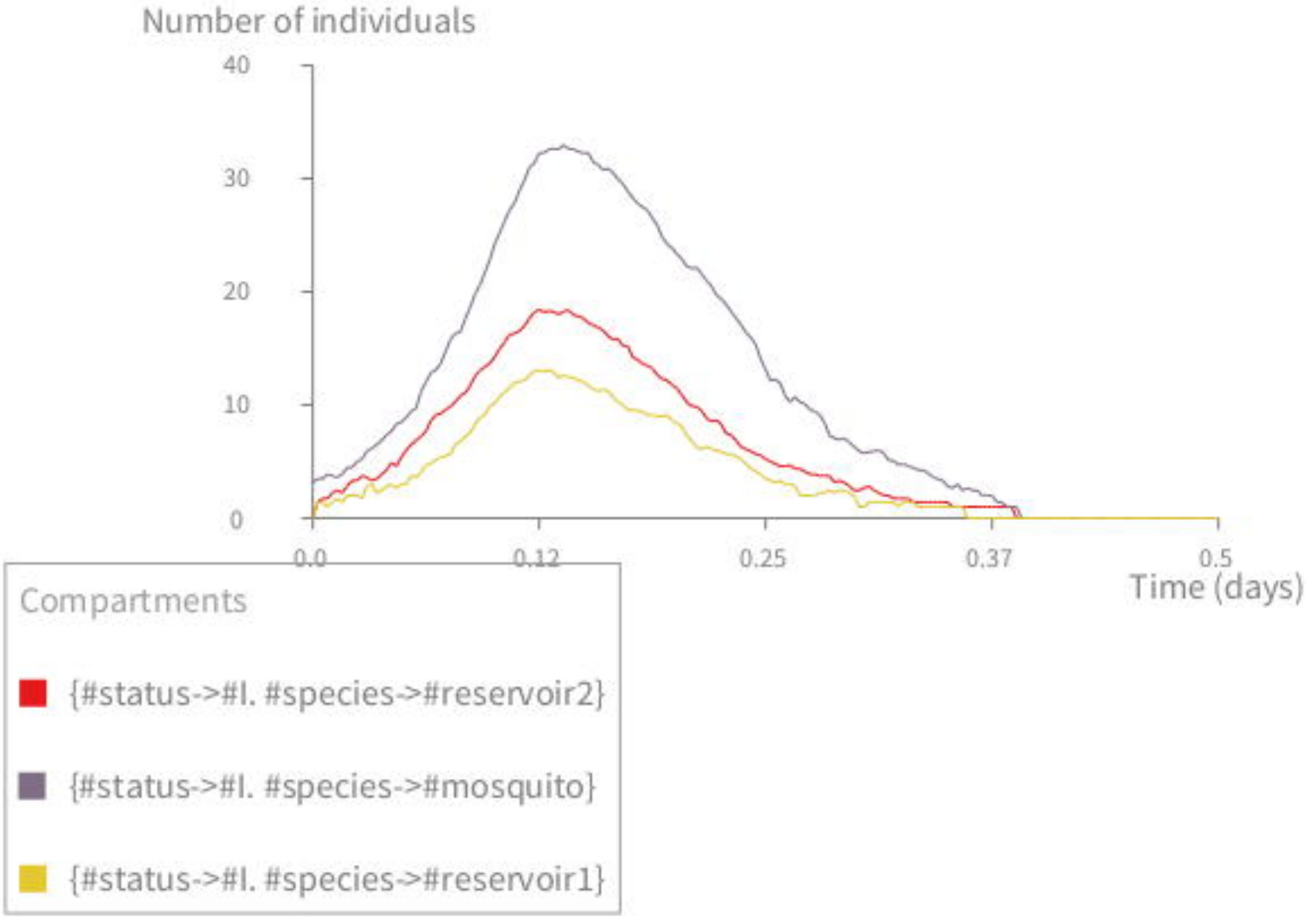
Code

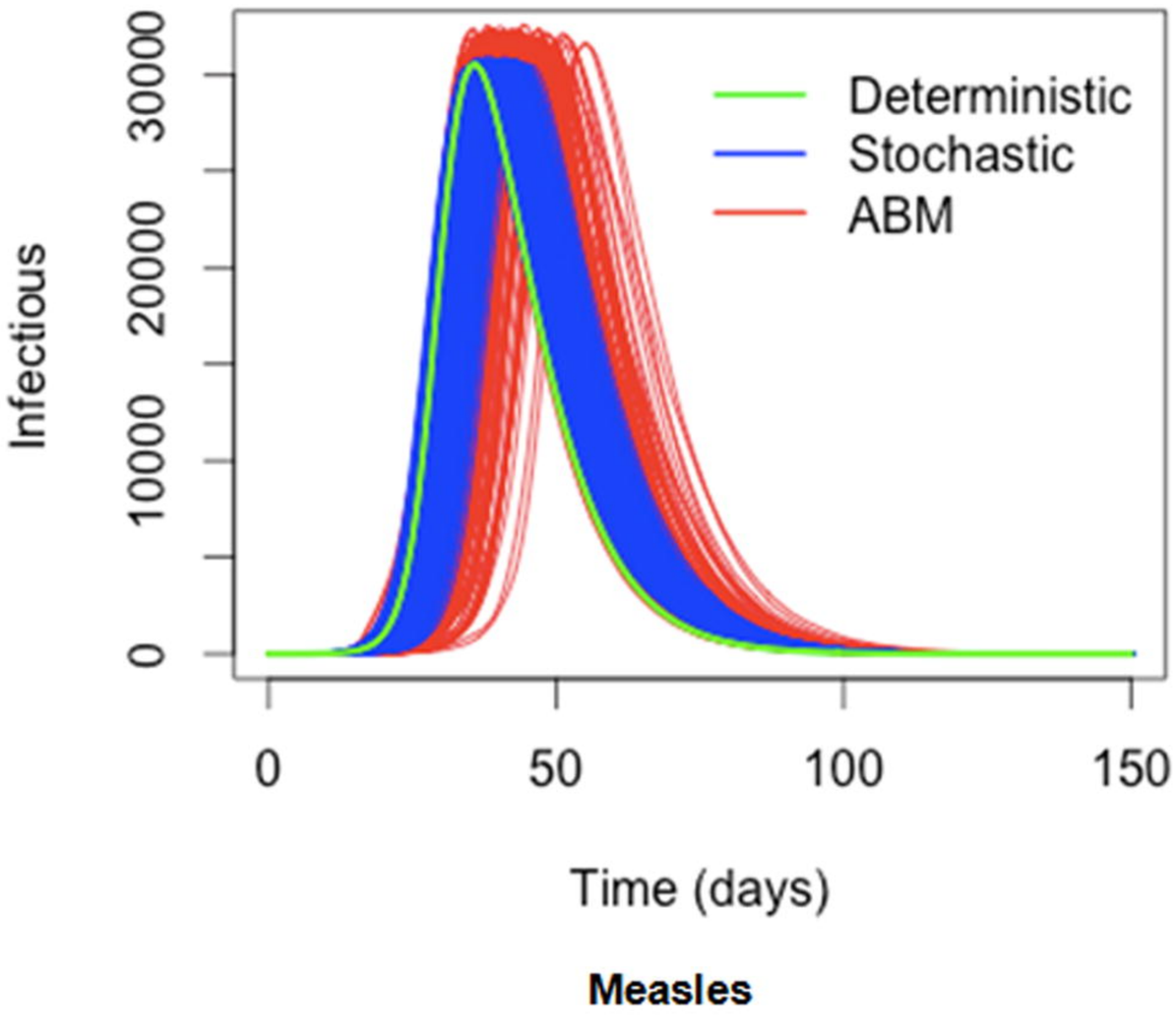
```
KendrickModel SIR
   attribute: #(status -> S I R);
   parameters: #(beta lambda gamma mu);
   transitions: #(
      S -- lambda --> I.
      I -- gamma --> R.
      status -- mu --> Empty.
      Empty -- mu --> S.
   ).

KendrickModel Species
   parameters: #(rho);
   attribute: #(species -> mosquito reservoir1 reservoir2).

Composition Mosquito
   model: 'SIR';
   model: 'Species'.

Scenario MosquitoPopulation
   on: 'Mosquito';
   populationSize: 13000;
   S_species: #(9999 1000 2000);
   I_species: #(1 0 0).

Scenario MosquitoParameters
   on: 'Mosquito';
   gamma: 52;
   beta: 1;
   mu_species: #(12.17 0.05 0.05);
   rho_species: #(
      #(0 0.02 0.02)
      #(0.02 0 0)
      #(0.02 0 0)
   );
   N: #(species);
   lambda: #(beta*rho*I sum).


Simulation MosquitoGillepsie gillespie
    scenarios: #(MosquitoPopulation MosquitoParameters);
   from: 0.0;
   to: 0.5;
   step: 0.0027.
```

TRCanvas

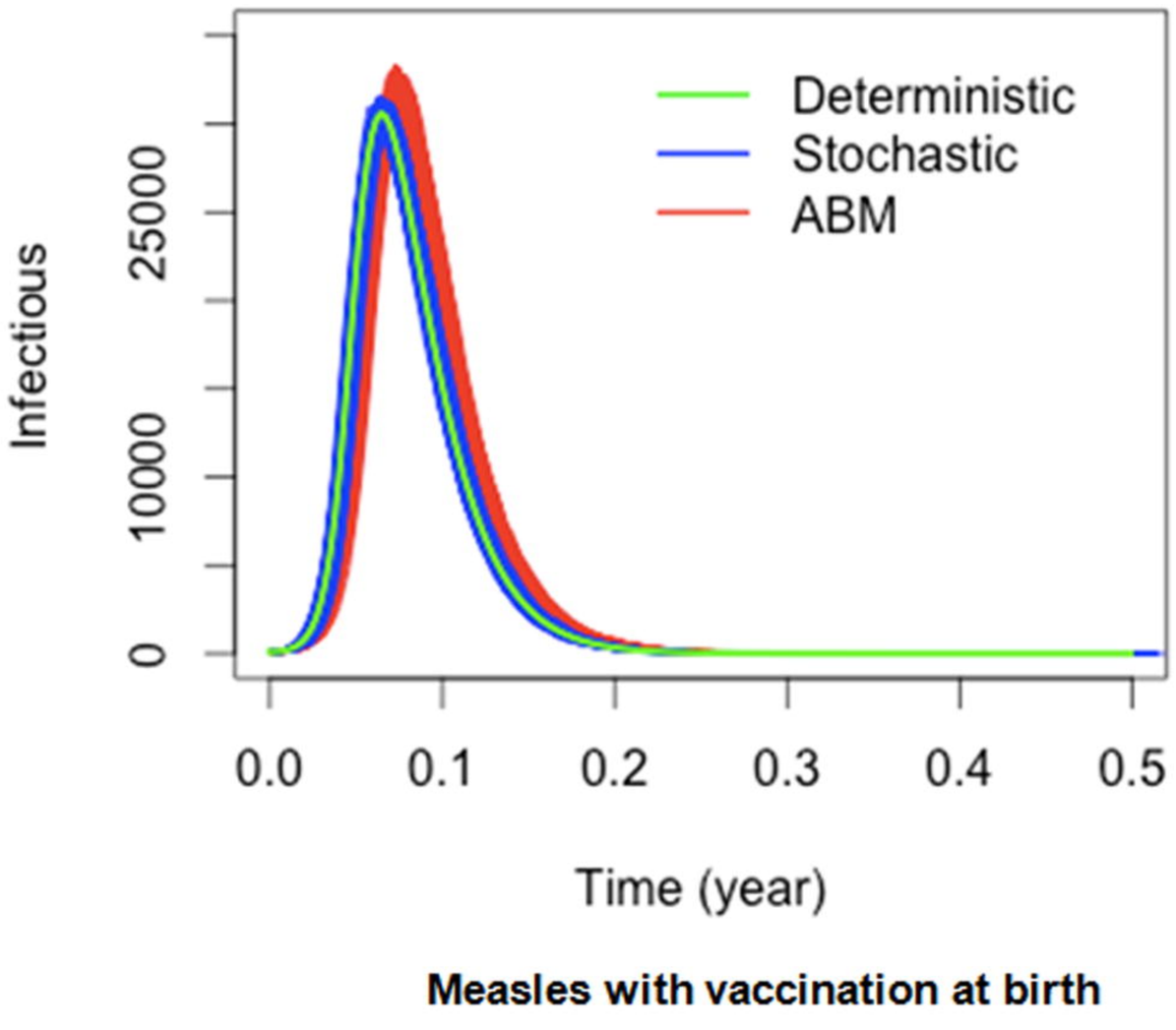
Canvas



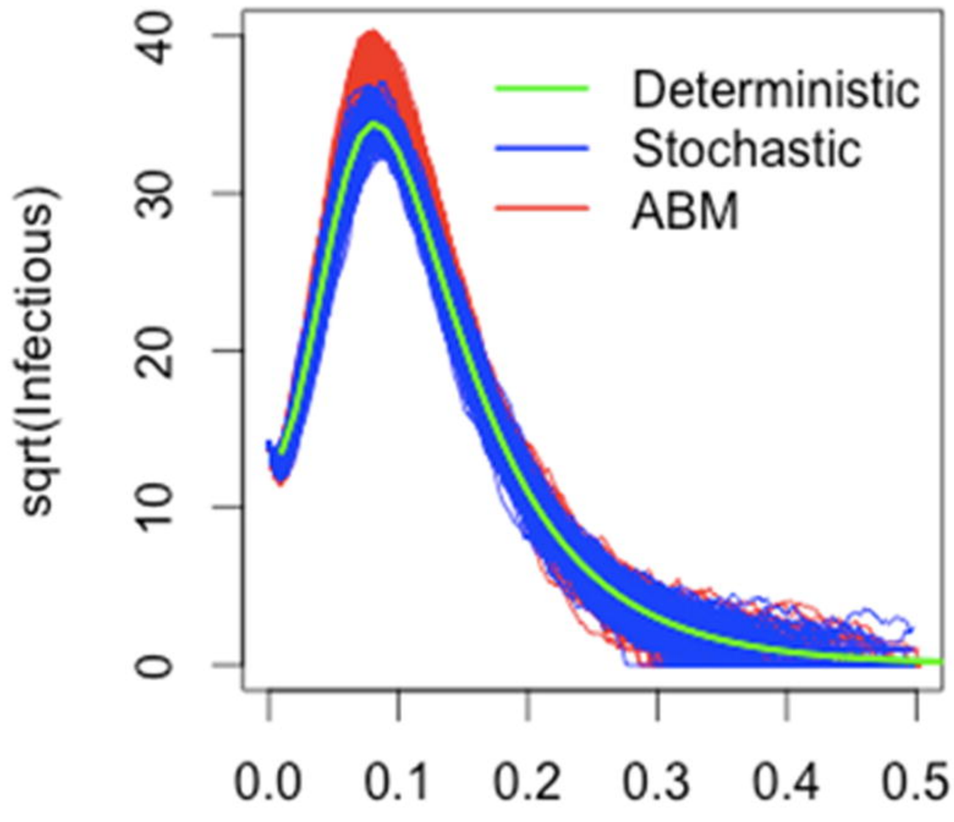

Number of individuals
40
30
20
10
0
0.0
0.12
0.25
0.37
0.5
Time (days)
Compartments
{#status->#I. #species->#reservoir2}
{#status->#I. #species->#mosquito}
{#status->#I. #species->#reservoir1}



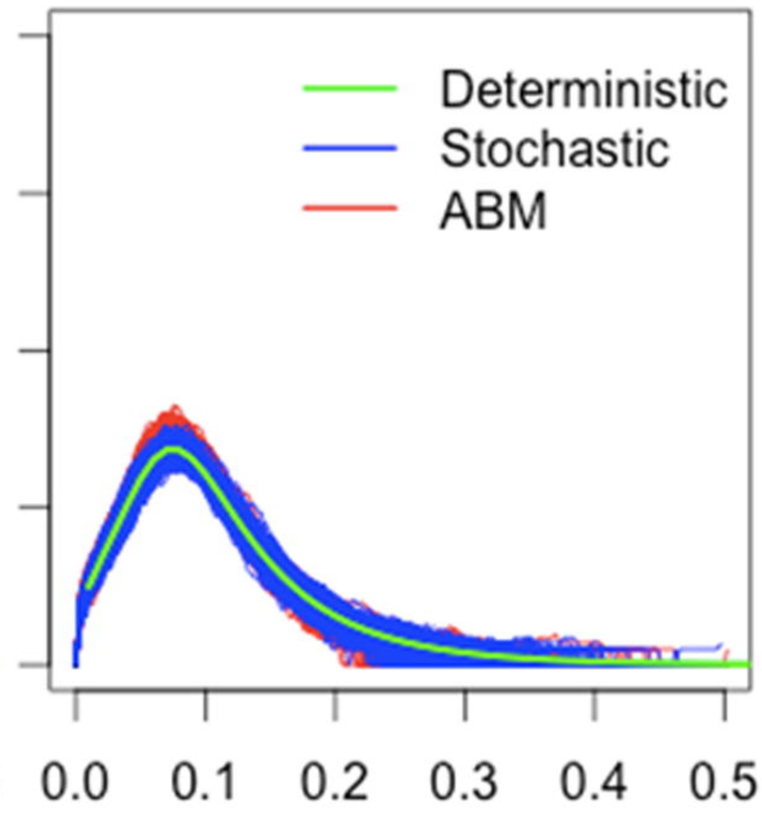
Measles



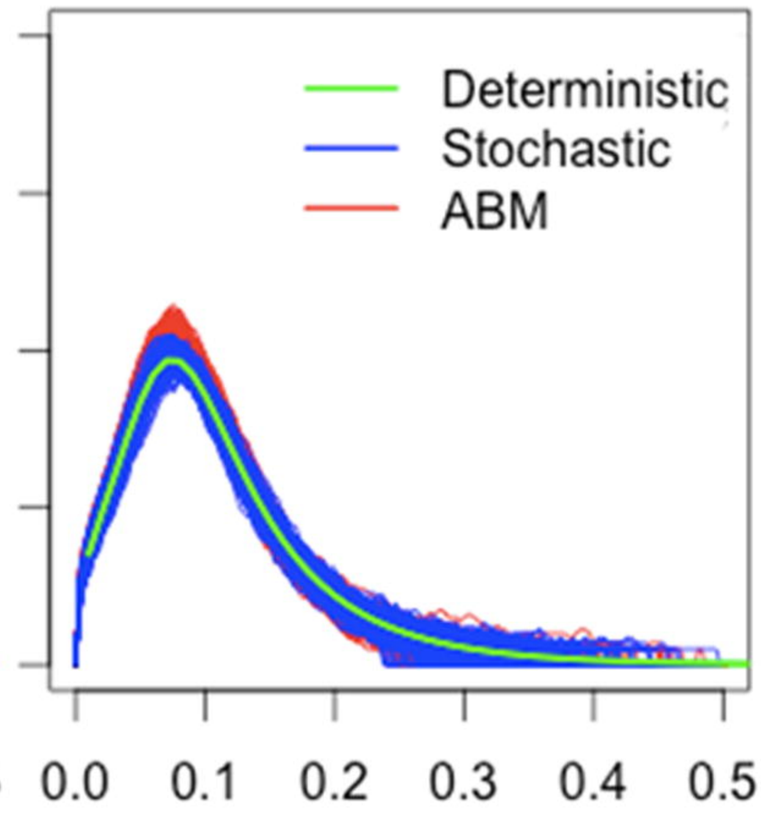
Measles with vaccination at birth



Mosquito



Reservoir1



Reservoir2