Chapter 1

Expressing Epidemiological Models

Understanding how infectious diseases propagate is a key challenge for the 21st century. Mathematical modeling is a powerful tool for studying complex systems that are commonly used in many scientific disciplines. It is widely used to model infectious diseases in order to study the mechanisms of transmission, explore characteristics of epidemics, predict the future course of an outbreak and evaluate strategies to find a best control-program like quarantine. The first mathematical model of epidemiology was proposed by Daniel Bernoulli in 1766 to defend the practice of inoculation against smallpox. The major contribution to modern mathematical epidemiology was carried out by Kermack and McKendrick who formulated a compartmental model based on relatively simple assumptions on the rates of flow between different classes categorized by epidemiological status.

Kendrick is an embedded domain-specific language in Pharo for defining mathematical models of epidemiology. This chapter analyzes and visualizes the spatiotemporal evolution of epidemiological models using Roassal.

This chapter was written with the participation of Bui Thi Mai Anh (maianht2@gmail.com), Serge Stinckwich (serge.stinckwich@ird.fr), and Nick Papoulias (npapoylias@gmail.com).

1.1 Compartmental models of epidemiology

The targeted models of the **Kendrick** language are the SIR or SEIR compartmental model in which each individual goes through three distinct phases:

- First an individual is considered to be *Susceptible* to pathogens (status

$$\begin{cases} \frac{dS}{dt} & = & -\beta SI \\ \frac{dI}{dt} & = & \beta SI - \gamma I \\ \frac{dR}{dt} & = & \gamma I \end{cases}$$

Figure 1.1: Mathematical description of SIR model using ODEs.

S),

- Once infected, the individual is *Infectious* (status I) and spreads the infection,
- After *Recovery* (status R), the individual is immunized and cannot become infected again.

The transition of status between compartments is represented mathematically as derivatives of compartment size with respect to time.

At the moment, **Kendrick** supports for the mathematical models of epidemiology based on ordinary differential equations (**ODEs**). The system of ODEs followed represents the SIR classic model of epidemiology (see figure 1.1).

These models are specified using the Kendrick domain-specific language and modeled using the simulation modules integrated into Kendrick platform. The simulator takes a Kendrick model (*i.e.,* an epidemiological model written in the Kendrick language) and performs a simulation algorithm and outputs the result showing the spatial and temporal evolution dynamics of each compartment. Roassal visualizes the result.

The simulation module supports three modeling formalisms: deterministic, stochastic and individual-based (also called agent-based). The modelers can switch between the simulation modes by indicating the algorithm used.

The deterministic simulation resolves the ODEs system of the model and produces numerical results. Kendrick uses deterministic solvers implemented in the package PolyMath, including RungeKutta and Euler. These solving methods are used to find numerical approximations to the solutions of ODEs. The results are always the same for all simulations under the predefined model parameters and initial conditions (initial values of compartments). Therefore, deterministic models reflect the average dynamics of the disease.

While deterministic models provide insights into the endemic equilibrium and its stability, shifting to stochastic models is known to be more realistic in unrderstanding and predicting the dynamics of diseases. Kendrick currently supports some Simulation Stochastic Algorithms (SSA) of Gillespie - converting the ODEs system of the model to stochastic events. After each time step, an event is randomly choosen to be executed. Kendrick allows one to increase the model accuracy using an individual-based simulator. Every individual of the population is examined at each time step and each individual has the probability of changing its current status.

The numerical results produced by simulations can be displayed graphically by using visualization modules of Kendrick. Such modules are implemented based on some features of Roassal.

There are three kinds of visualizations: diagrams (using RTGrapher of Roassal), maps (using RTMapBuilder) and networks (using RTMondrian).

1.2 Visualizing Ebola epidemic outbreak from data

The website https://github.com/cmrivers/ebola provides data that represents the 2014 Ebola outbreak in West African countries. The following script visualizes data for Guinea and Liberia (Figure 1.2):

```
   tab := RTTabTable new input: (ZnEasy get:
'https://raw.githubusercontent.com/cmrivers/ebola/master/country_timeseries.csv')
   contents usingDelimiter: $,.
tab removeFirstRow.
tab replaceEmptyValuesWith: '0' inColumns: (2 to: 12) asArray.
tab convertColumnsAsInteger: (2 to: 12) asArray.
tab convertColumnAsDateAndTime: 1.
tab replaceZeroWithCumulativeValuesInColumns: (2 to: 12) asArray.
data := tab values reversed.

minValue := data minValue: [ :aData | aData first julianDayNumber ].

b := RTGrapher new.
b extent: 400@200.
ds := RTData new.
ds noDot.
ds connectColor: Color blue.
ds points: data.
ds y: [ :v | v at: 3 ].
ds x: [ :v | v first julianDayNumber − minValue ].
b add: ds.

ds := RTData new.
ds noDot.
ds connectColor: Color green.
```

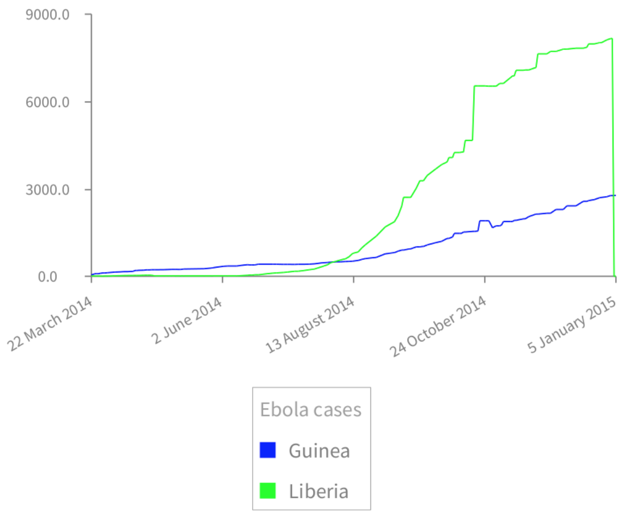

Figure 1.2: Visualizing Ebola cases in two African countries.

```
ds points: data.
ds y: [ :v | v at: 4 ].
ds x: [ :v | v first julianDayNumber − minValue ].
b add: ds.

b axisX
   labelRotation: −30;
   labelConversion: [ :v | (Date julianDayNumber: v + minValue) ].
b build.

lb := RTLegendBuilder new.
lb view: b view.
lb addText: 'Ebola cases'.
lb addColor: Color blue text: 'Guinea'.
lb addColor: Color green text: 'Liberia'.
lb build.
b view
```

1.3 Loading Kendrick

Kendrick is loadable using a Roassal plugin (Figure 1.3):

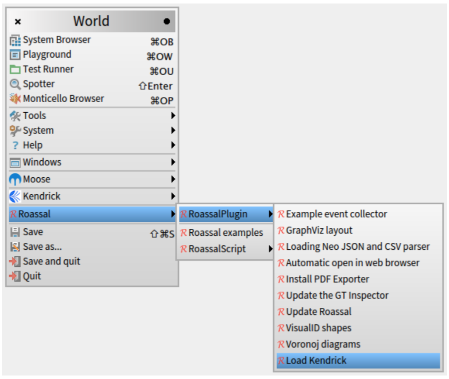

Figure 1.3: Loading Kendrick.

1.4 Measles model

Measles is a well-known childhood infectious disease. The most appropriate model representing measles epidemics is the SEIR model where individuals are categorized to be in four classes: first, all newborn individuals are assumed to be in the *Susceptible* (S) class at birth rate μ, then enter in *Exposed* (E) class who are infected but not yet infectious with transmission rate β, then become *Infectious* (I) after a latent period given by $1/\sigma$, and finally change to *Recovery* after an infectious period $1/\gamma$. The parameter N represents the total size of the population. In this example, the diagrams will be used to visualize the dynamics of the infectious disease. The following script illustrates the SEIR model of measles (Figure 1.4):

```
model := KEModel new population: (KEPopulation size: 100000).
model addAttribute: #status value: #(S E I R).
model atCompartment: { #status->#S } put: 99999 atOthersPut: 0.
model atCompartment: { #status->#I } put: 1.
model addEquations: {
        'S:t=mu*N - beta*S*I - mu*S'.
        'E:t=beta*S*I - sigma*E - mu*E'.
        'I:t=sigma*E - gamma*I - mu*I'.
        'R:t=gamma*I - mu*R'
        }.
model addParameters: {
      #beta->0.0000214.
      #gamma->0.143.
      #mu->0.0000351.
      #sigma->0.125 }.
simulator := KESimulator new: #RungeKutta from: 0.0 to: 150 step: 1.
simulator executeOn: model.
```

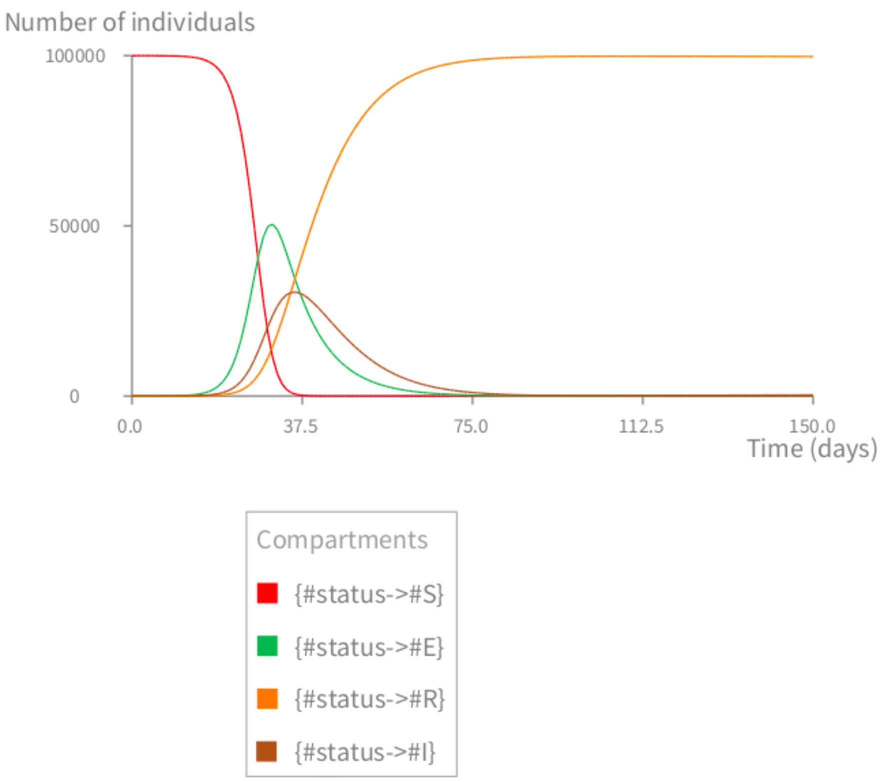

Figure 1.4: Modeling the Measles model with Kendrick.

```
diag := (KEDiagramBuilder new) data: simulator allTimeSeries.
diag xLabel: 'Time (days)'.
diag open
```

The rates mentioned of the model are introduced as parameters. This model is run in deterministic simulation in a period of 150 days (using the classical Runge–Kutta method, also known as RK4) with an initial population of 100000 individuals. Figure 1.4 shows the temporal dynamics of all compartments S, E, I and R. The Y axis counts the number of individuals in each class and the X axis is the time period of simulation (in days).

The figure shows that after about 37 days, the endemic reaches the peak, then the number of infectious reduces, and finally the number of recovered individuals increases.

$$\begin{cases} \frac{dS_i}{dt} &= \mu_i N_i - \sum_j \beta_{ij} I_j S_i - \mu_i S_i \\ \frac{dI_i}{dt} &= \sum_j \beta_{ij} I_j S_i - \sigma_i I_i - \mu_i I_i \\ \frac{dR_i}{dt} &= \sigma_i I_i - \mu_i R_i \end{cases}$$

Figure 1.5: Mathematical description of the multi-hosts model using ODEs.

1.5 Complex models in Kendrick

The modeling of infectious disease involves not only the study of their transmission cycle (as shown in the previous example) but also the effects of the population heterogeneities due to age intervals, sexes, species, viral strains or spatial regions etc. We consider such heterogeneities as concerns of epidemiology. The Kendrick DSL allows one to construct a model as a composition of modular models that describe specific epidemiological concerns. Each concern is separately defined and then integrated into the model. Look at the following examples.

1.6 Example of multi-host model

In many situations, due to the fact that the primary hosts do not interact directly to transmit infection, the spreading of diseases requires a secondary host (vector-borne diseases). In this example, we investigate the mosquito-borne disease with two sources. Here, the matrix of transmission $beta$ indicates that the disease is only transmitted between mosquitos and two reservoirs, so that the main diagonal of this matrix is zero. A reservoir is the long-term host of a pathogen of an infectious disease. Example of natural reservoirs for pathogens are: rats or bats. This disease is mathematically formulated as shown in figure 1.5.

We specify the multi-host aspect of epidemiology as a concern (called multi-host concern) and then apply it to the model.

The Kendrick model of this disease is specified as follows:

```
multiHostConcern := KEConcern new.
multiHostConcern
   addAttribute: #species
   value: #(#mosquito #reservoir1 #reservoir2).

model := KEModel new.
model population: (KEPopulation size: 13000).
model addAttribute: #status value: #(#S #I #R).
model addParameters: {
      #beta –> 1.
      #gamma –> 52.
      #mu –> 12.17.
}.
model addParameter: #lambda value: 'beta*I'.
model addEquations: {
   'S:t=mu*N – lambda*S – mu*S'.
   'I:t=lambda*S – gamma*I – mu*I'.
   'R:t=gamma*I – mu*R'
}.

model integrate: multiHostConcern.

model
   atParameter: #mu
   assignValue:
   [ :aModel| |c val|
      c := aModel currentCompartment at: #species.
      c = #mosquito ifTrue: [ val := 12.17 ].
      c = #reservoir1 ifTrue: [ val := 0.05 ].
      c = #reservoir2 ifTrue: [ val := 0.05 ].
      val
   ].
model
   atParameter: #N
   assignValue:
   [ :aModel| |c|
      c := aModel currentCompartment at: #species.
      aModel sizeOfPopulation: c
   ].
model
   atParameter: #beta
   assignValue:
   [ :aModel| |c val|
      c := aModel currentCompartment at: #species.
      c = #mosquito ifTrue: [ val := #(0 0.02 0.02) ].
      c = #reservoir1 ifTrue: [ val := #(0.02 0 0) ].
      c = #reservoir2 ifTrue: [ val := #(0.02 0 0) ].
      val
   ].
model
   atParameter: #lambda
   assignValue:
   [ :aModel|
      ((aModel atParameter: #beta) *
      (aModel atCompartment: {#status–>#I})) sum
   ].
```

```
model atCompartment: { #status−>#S. #species−>#mosquito } put: 9999.
model atCompartment: { #status−>#I. #species−>#mosquito } put: 1.
model atCompartment: { #status−>#S. #species−>#reservoir1 } put: 1000.
model atCompartment: { #status−>#S. #species−>#reservoir2 } put: 2000.

simulator := KESimulator new: #Gillespie from: 0.0 to: 0.5 step: 0.0027.
simulator executeOn: model.
db := (KEDiagramBuilder new) data: ((simulator timeSeriesAt: '{#status: #I}') sqrt).

db xLabel: 'Time (year)'.
db yLabel: 'sqrt(Infectious)'.
db open
```

In this example, we specify the multi-host concern separately from the SIR model. This concern contains no parameters. Its attribute *species* contains three values for mosquito, reservoir1, reservoir2. When integrating this concern to the SIR model, the population is further decomposed in 3x3 compartments. Because of the heterogeneities caused by the multi-host concern, parameters of SIR model such as β, μ, λ, N (N by default is the size of the population) become heterogeneous. For example, according to the species, the parameter μ may have different values (12.17 for mosquito, 0.05 for reservoir1 and reservoir2). Such parameters will be re-defined as shown in the script (by re-assigning values).

In this example, we demonstrate the use of stochastic simulation (by indicating the used algorithm as #Gillespie). The temporal evolution of this disease is shown in the Figure 1.6. The diagram illustrates the dynamics of the infectious compartment for each species. The Y-axis is squared for readability.

1.7 An example of Ebola spatial model with Kendrick

This example is used for demonstrating the map visualization of Kendrick. We will study how a spatial concern is defined with Kendrick.

The example below describes the Ebola model in six countries of Africa. Due to the characteristic of the Ebola epidemic, the infection transmission only occurs via direct contact between the Susceptibles and an Infectious. Hence, we consider that the spatial affects the transmission of infection by the migration of infectious individuals from a country to another country.

In this example, we specify two concerns separately: the spatial concern and the SIR concern, then apply them to the model. The script of the model is below:

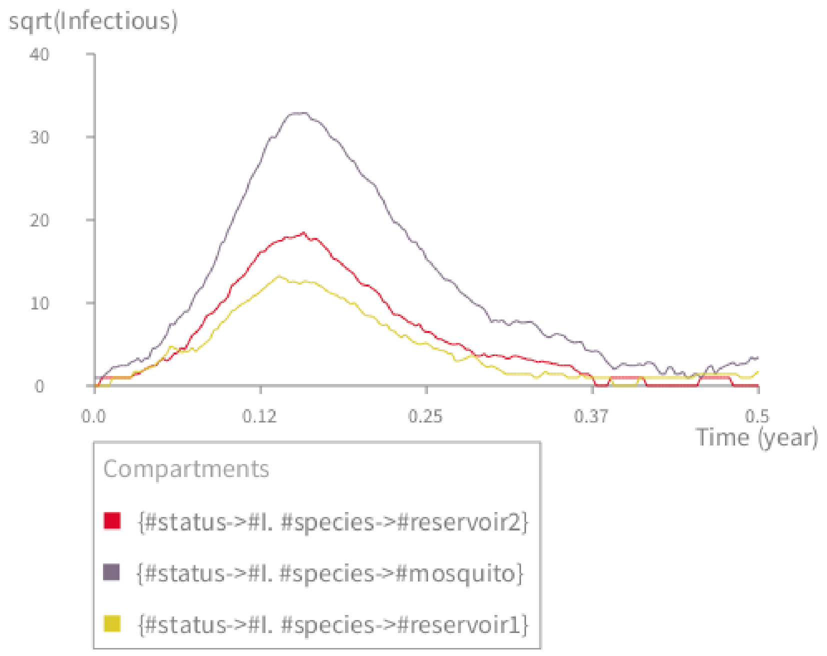

Figure 1.6: Modeling Mosquito-borne model with Kendrick.

```
model := KEModel new population: (KEPopulation size: 6000).

map := KEMap new.
map countries: #(#Liberia #Guinea #SierraLeone #Nigeria #Senegal #Niger).
map routesFrom: #Liberia toAll: #(#Guinea).
map routesFrom: #Guinea toAll: #(#SierraLeone).
map routesFrom: #SierraLeone toAll: #(#Nigeria).
map routesFrom: #Nigeria toAll: #(#Senegal).
map routesFrom: #Senegal toAll: #(#Niger).
map routesFrom: #Niger toAll: #(Liberia).
spatialConcern := KEConcern new.
spatialConcern addAttribute: #country value: map countries.
spatialConcern addParameter: #rho value: 0.05.
spatialConcern transitions: (map routesToTransitions: 'rho').

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: #(#lambda #beta #gamma).
sirConcern
   addTransitionFrom: { #status−>#S }
   to: { #status−>#I }
   probability: 'lambda'.
sirConcern
   addTransitionFrom: { #status−>#I }
   to: { #status−>#R }
   probability: 'gamma'.

model integrate: sirConcern.
model integrate: spatialConcern.

model
   atParameter: #beta
   assignValue: 0.0002.
model
   atParameter: #gamma
   assignValue: 0.1.
model
   atParameter: #lambda
   assignValue:
   [ :aModel| |c|
      c := aModel currentCompartment at: #country.
      (aModel atParameter: #beta)*(aModel atCompartment: {#status−>#I. #country−
      >c}) ].

#(#Guinea #SierraLeone #Nigeria #Senegal #Niger) do: [ :c|
   (model atCompartment: { #status−>#S. #country−>c } put: 1000)
].
model atCompartment: { #status−>#S. #country−>#Liberia } put: 950.
model atCompartment: { #status−>#I. #country−>#Liberia } put: 50.

simulator := KESimulator new: #IBM from: 0 to: 100 step: 0.1.
simulator executeOn: model.

mapBuilder := KEMapBuilder africa.
mapBuilder
   data:
   (mapBuilder countries collect: [:c|
      ((model atAttribute: #country) includes: c)
      ifTrue: [
         (simulator timeSeriesAt: {#status−>#I. #country−>c}) first peakOfEpidemic ]
      ifFalse: [ 0 ]
   ]).
mapBuilder open.
diagBuilder := KEDiagramBuilder new data: (simulator timeSeriesAt: {#status−>#I}).
diagBuilder open.
```

Figure 1.7 represents the results of the individual-based simulation on six countries. The left hand diagram shows the evolution dynamics during 100

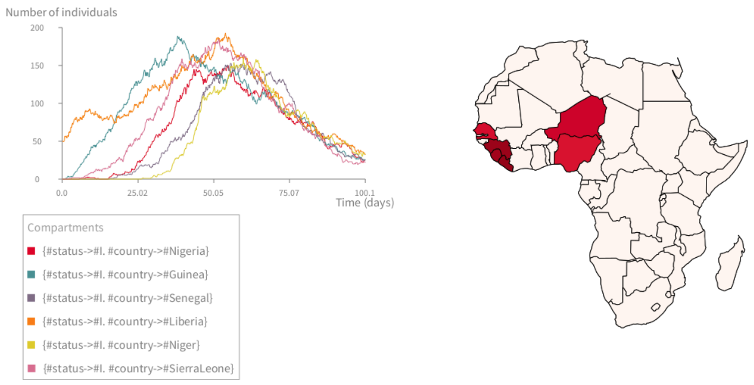

Figure 1.7: Representing the Ebola disease with Kendrick in 6 countries of Africa.

days. The right hand graph visualizes six countries in Africa, the color of each one depends on its infection degree.

1.8 Network Visualization of Epidemiological Models

In all examples above, individuals of the population are assumed to mix randomly and uniformly. Each individual therefore has an equal probability to have contact with all other individuals. But in fact, the number of contacts an individual has is considerably smaller than the size of the population. The network model supposes a network of contacts between individuals of the population. It assigns to each individual a finite set of contacts. Thus, the probability of infection of a susceptible individual depends on the number of infectious contacts he or she has.

Models that incorporate network structure of epidemiology are captured by Kendrick as a spatial concern. The network is represented through graphs in which a node represents an individual (or a set of individuals having the same contacts) and an edge between two nodes represents an interaction that may allow disease transmission (contact). Constructing a real contact network required knowledge of every individual in a population and every disease-causing contact between them. This is typically unfeasible for even small populations. Thus people often work with approximate networks, particularly computer-generated networks. Kendrick supports some computer-generated networks that have been investigated in the domain of epidemiology including the Poisson random network, small-world network, and scale-free network.

The visualization of network in Kendrick is done by the network builder module. The example below represents a random network of a population of 100 individuals. Each node of the network corresponds to an individual of the population. The model is run as an individual-based simulation.

```
model := KEModel new population: (KEPopulation size: 100).

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: { #beta. #gamma. #lambda }.
sirConcern
   addTransitionFrom: { #status−>#S }
   to: { #status−>#I }
   probability: 'lambda'.
sirConcern
   addTransitionFrom: { #status−>#I }
   to: { #status−>#R }
   probability: 'gamma'.

network := KEContactNetwork nodes: 100 topology: { #random. #p−>0.02 }.
spatialConcern := KEConcern new.
spatialConcern addParameter: #network value: network.
spatialConcern addAttribute: #node value: network allContacts.

model integrate: sirConcern.
model integrate: spatialConcern.

model
   atParameter: #lambda
   assignValue:
   [ :aModel||node|
      node := aModel currentCompartment at: #node.
      ((aModel atParameter: #network)
         contactsOf: {aModel. #node−>node. #status−>#I})
      * (aModel atParameter: #beta)/(aModel atParameter: #N)
 ].
model atParameter: #beta assignValue: 100.
model atParameter: #gamma assignValue: 0.1.

1 to: 99 do: [:i|
   model atCompartment: {#status−>#S. #node−>i asString asSymbol} put: 1].
model atCompartment: { #status−>#I. #node−>#'100' } put: 1.

simulator := KESimulator new: #IBM from: 0.0 to: 50 step: 0.1.
simulator executeOn: model.
nb := KENetworkBuilder new
      data: simulator allTimeSeries;
      network: (model atParameter: #network);
```

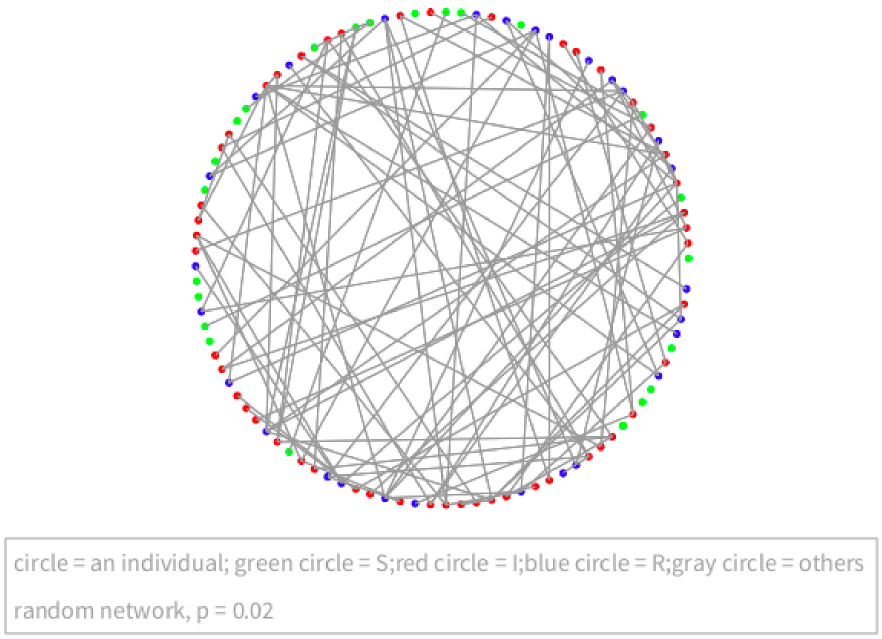
circle = an individual; green circle = S;red circle = I;blue circle = R;gray circle = others
random network, p = 0.02

Figure 1.8: Visualizing a random network population with Kendrick.

```
        status: #(#S #I #R);
        colors: #(#green #red #blue);
        viewDataAtTime: 10;
        legend: 'random network, p = 0.02'.
nb open
```

We visualize the network of contacts after 10 days. The random network is characterized by the probability of having an edge between two random nodes (the argument p). The random network represents the contacts between individuals of the model and can be seen in Figure 1.8.

We may change the topology of the network to examine the effects of other kind networks. For example, defining a scale-free network of 100 nodes with the first number of edges is m0:

```
    network := KEContactNetwork nodes: 100 topology: { #scalefree. #m0−>1 }.
```

The scale free network of the model after 10 days can be seen in Figure 1.9.

As can be seen on these figures, the random network shows a lack of clustering because each two random nodes have the same probability to make

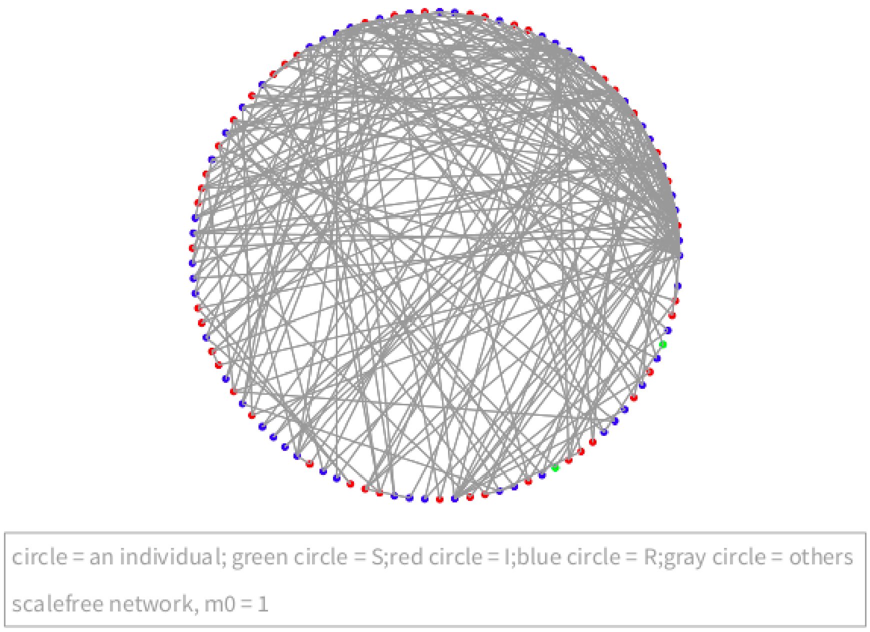
circle = an individual; green circle = S;red circle = I;blue circle = R;gray circle = others
scalefree network, m0 = 1

Figure 1.9: Visualizing a scale-free network population with Kendrick.

an edge. In the scale-free network, some individuals may have many more contacts than others. The smallworld network is formed from a lattice grid with some rewire nodes. This network is defined below:

```
network := KEContactNetwork nodes: 100 topology: { #smallworld. #K−>2. #beta
   −>0 }.
```

Beta is the probability to rewire an edge in the network; K is the number of contacts on two sides of a node. By making K = 2 and beta = 0, we can create a much simpler model like the ring network as shown in Figure 1.10. Such kinds of networks are often studied in the context of epidemiology to show the social contacts between individuals (i.e, in the case of sexually transmitted diseases).

As mentioned above, a node of a contact network may represent a group of individuals with some contact between one another. In such cases, in a node, an individual is supposed to contact all others of the same node. The probability of infection of a susceptible depends on the number of infectious contacts within its node and in other linked nodes.

The example below represents a contact network of 10 nodes defined on

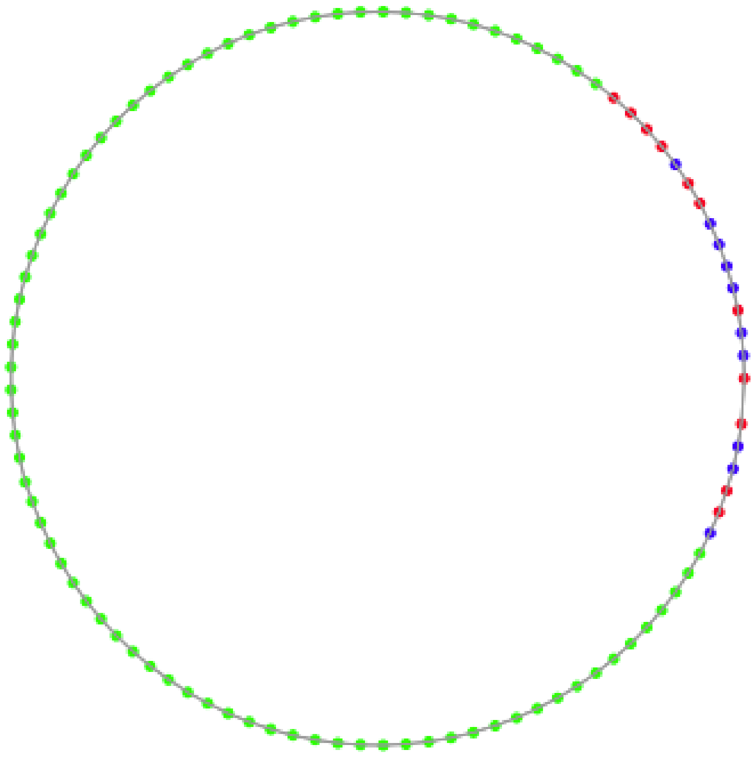

Figure 1.10: Visualizing a small-world network population with Kendrick.

a population of 1,000 individuals. The visualization of the network can be seen in Figure 1.11. We showed the infectious prevalence of each node (as the size of nodes).

```
model := KEModel new population: (KEPopulation size: 1000).

sirConcern := KEConcern new.
sirConcern addAttribute: #status value: #(S I R).
sirConcern addParameters: { #beta. #gamma. #lambda }.
sirConcern
   addTransitionFrom: { #status−>#S }
   to: { #status−>#I }
   probability: 'lambda'.
sirConcern
   addTransitionFrom: { #status−>#I }
   to: { #status−>#R }
   probability: 'gamma'.
```

```
spatialConcern := KEConcern new.
network := KEContactNetwork nodes: 10 topology: { #random. #p−>0.2 }.
spatialConcern addParameter: #network value: network.
spatialConcern addAttribute: #node value: network allContacts.

model integrate: sirConcern.
model integrate: spatialConcern.

model
   atParameter: #lambda
   assignValue:
   [ :aModel||node|
      node := aModel currentCompartment at: #node.
      ((aModel atParameter: #network)
         contactsOf: {aModel. #node−>node. #status−>#I})
      * (aModel atParameter: #beta)/(aModel atParameter: #N)
 ].
model atParameter: #beta assignValue: 1.
model atParameter: #gamma assignValue: 0.1.

2 to: 10 do: [:i|
   model atCompartment: {#status−>#S. #node−>i asString asSymbol} put: 100].
model atCompartment: { #status−>#I. #node−>#'1' } put: 1.
model atCompartment: { #status−>#S. #node−>#'1' } put: 99.

simulator := KESimulator new: #IBM from: 0.0 to: 100 step: 0.1.
simulator executeOn: model.
nb := KENetworkBuilder new
      data: simulator allTimeSeries;
      network: (model atParameter: #network);
      status: #(I); viewDataAtTime: 30.
nb open
```

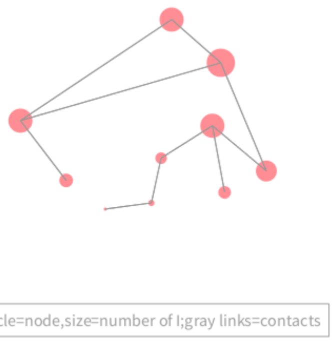

Figure 1.11: Visualizing a network contact model between groups of individuals with Kendrick.