Chapter 8

Online Appendix

Winfried Just[1], Hannah Callender[2], M. Drew LaMar[3] and Natalia Toporikova[4]
[1]Ohio University, Athens, OH, USA, [2]University of Portland, Portland, OR, USA, [3]The College of William and Mary, Williamsburg, VA, USA, [4]Washington and Lee University, Lexington, VA, USA

8.1 OUR NETLOGO CODE IONTW

We will be using NETLOGO to run simulations of our models. NETLOGO is a software platform for exploration of complex systems. If you would like to know more about the software, you can check out the NETLOGO website [1], as well as the Wikipedia entry for NETLOGO [2]. Among the resources available at the NETLOGO website are tutorials and a full software documentation. If NETLOGO is not yet installed on your computer, you will first need to download it from the NETLOGO website mentioned above before proceeding.

NETLOGO is a user-friendly software designed to teach concepts of programming and allow users to visualize simulations through the use of agent-based modeling. It allows the user to create dynamic agent-based models, visualize these models over the course of time, and make predictions based on user-defined inputs and model outputs. We have created a NETLOGO program that we call IONTW, which is an acronym for "Infections On NeTWorks." IONTW will serve as our experimental tool for modeling and making predictions about the spread of infectious diseases under a variety of different conditions and assumptions, in particular, assumptions about certain control strategies.

A complete description of IONTW can be found at our website [3]. For convenience, we include here a package IONTWv1.1.zip that contains the following materials that have been adapted from this source:

- Version 1.1 of the code IONTW
- A reference guide on using IONTW
- Sample input files for the **Load** option of IONTW

After installing NETLOGO on your computer, download IONTWv1.1.zip and extract these files into a directory that you will be using for all simulation exercises. Then familiarize yourself with the reference guide and the introductory material that you will find on our website [3].

8.2 WAIT! YOUR ANSWER IS DIFFERENT THAN MINE: INHERENT VARIABILITY IN NETLOGO SIMULATIONS

In this section, we provide an introduction to the use of our NETLOGO code IONTW for modeling the spread of infectious diseases. Specifically, we will explore the variability in outputs of a model for the same sets of parameters and utilize the Central Limit Theorem to provide information on the number of runs needed to obtain confident estimates of the average outputs.

Let's first get familiar with IONTW. Install the code and familiarize yourself with it as instructed in Section 8.1.

Algebraic and Discrete Mathematical Methods for Modern Biology. http://dx.doi.org/10.1016/B978-0-12-801213-0.00027-7

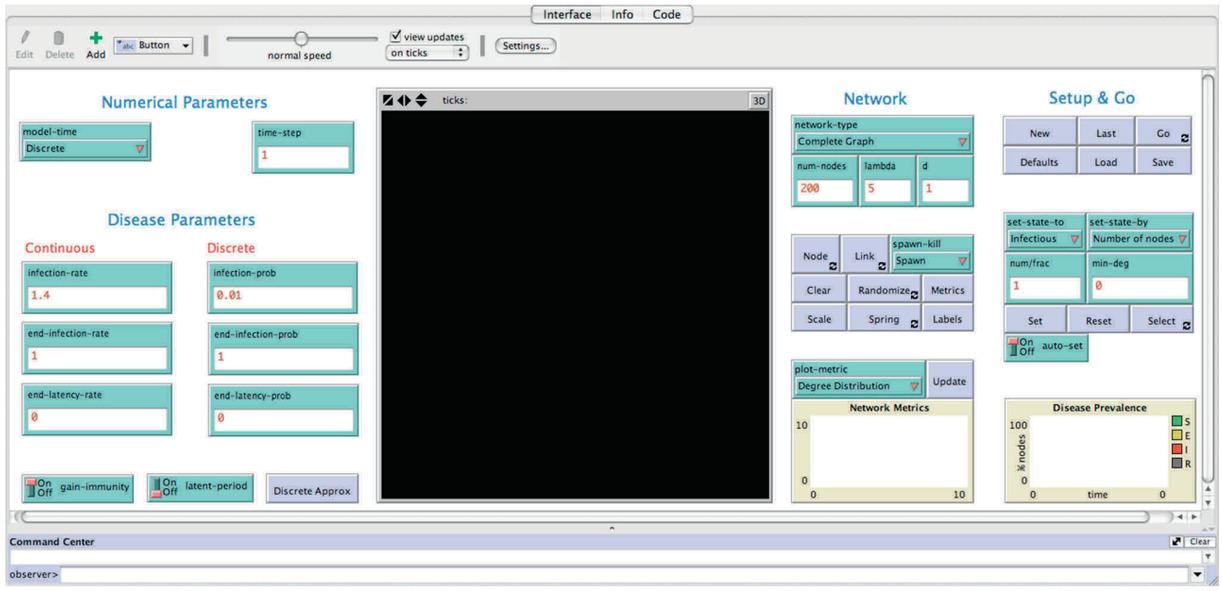

FIGURE 8.1 Opening window of IONTW.

Getting Ready

- Open the file IONTW. You should see a NetLogo window similar to the one in Figure 8.1. There are many options here to change certain parameters and other model settings. We will discuss a few of these settings in this section. You will learn about other settings in later sections and in the modules for Chapter 9 [4].
- Throughout this section we will keep all settings fixed. If your settings are not already set to the values below, change them before moving on. The settings that are not explicitly listed below are irrelevant for our work in this section. In all NetLogo exercises in this and the follow-up Chapter 9, we will only specify settings that are relevant for the given exercise.
  - **model-time → Continuous**
    This tells IONTW how to treat time in the simulation.
  - **infection-rate:** 0.02
    This is the parameter β, the rate at which two "average" hosts make effective contact.
  - **end-infection-rate:** 0.075
    This is the parameter α, the removal rate of "average" hosts. It is equal to $1/\langle \tau^I \rangle$, the reciprocal of the mean duration of infectiousness.
  - **gain-immunity: On**
    If set to **On**, this causes hosts to acquire immunity upon recovery from infection, thus simulating an *SIR*- or an *SEIR*-model. If set to **Off**, this causes hosts to become susceptible again, thus simulating an *SIS*- or *SEIS*-model.
  - **latent-period: Off**
    If set to **On**, this adds an **E**-compartment, thus simulating an *SEIR*- or an *SEIS*-model. If set to **Off**, this suppresses the **E**-compartment, thus simulating an *SIR*- or *SIS*-model.
  - **network-type → Complete Graph**
    This provides the structure of interactions among hosts and will be investigated further in Chapter 9.
  - **num-nodes:** 10
    This sets the total number N of hosts.
  - **set-state-to → Infectious**
  - **set-state-by → Number of nodes**

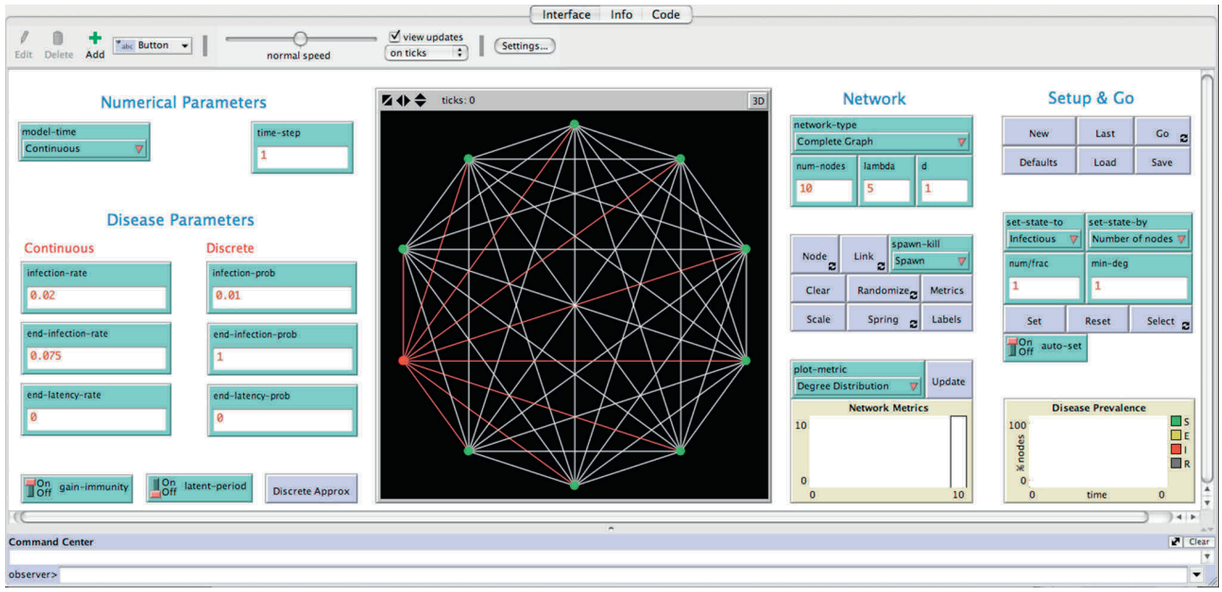

FIGURE 8.2 Window after clicking **New** in IONTW.

- **num/frac:** 1
- **min-deg:** 0
- **auto-set: On**
  The last five settings control how IONTW determines the initial state. Details will be discussed in Chapter 9. The settings specified above will always produce an initial state with exactly one infectious host in an otherwise susceptible population.

- It is best to save these settings once you have entered them, in case you want to complete this exercise in more than one session. To do so, simply select **File** → **Save As** from the toolbar, and save your file under an appropriate name.
- Next you need to initialize the simulation by clicking on **New** in the upper right-hand corner. You should see something like the picture in Figure 8.2.

The **World** window in the middle with the black background will show individual hosts (represented by colored dots) that will change color according to their current state. Green dots represent susceptible hosts, red dots represent infectious hosts, and gray dots represent removed hosts (i.e., hosts that have either gained immunity or died). The lines connecting one dot to another dot indicate a possible direct contact between the hosts represented by these dots, thus allowing an opportunity for transmission of the disease. Notice that the red lines connect infectious with susceptible hosts; these are the ones that permit a successful contact at the current stage of the simulation.

To the right of the **World** window, you have already noticed a tab labeled **network-type**. We will investigate this tab further in Chapter 9. For now we will only use the **Complete Graph** option, which for our purposes means that each host can have direct contact with all other hosts.

The plot labeled **Disease Prevalence** displays a summary of the information in the **World** window. It records the percentage of hosts that are infectious (in red), susceptible (in green), and removed (in gray) as a function of time.

- Before running your first simulation, initialize the state of the population again by clicking on **New** a second time. What happens in the **World** window? Are there any differences? Try this several times and observe differences in the initial state of each host. Why might these changes occur? Is this what you would expect? Why or why not?

Experiment 1: Running a Simulation

You are now ready to run your first simulation. Make sure you have followed all of the previous steps, including setting the correct parameter values listed above and then clicking on **New**. At the top of the window you will also see a slider that is likely currently set to **normal speed**. This controls the speed at which you can visualize the simulation. To slow down a simulation, you can move the slider to the left. To speed it up, move the slider to the right. This even works while the simulation is running.

- Before running your first simulation, move the speed slider half way between the slowest setting and the default **normal speed** setting. This will allow you to better visualize the simulations for this section.
- To start the simulation, click **Go**.
- The simulation will run until there are no more infectious hosts, that is, until all dots are green or gray. If you wish to terminate a simulation before this happens, merely click on the **Go** button again. To resume the same simulation from where you stopped it, click **Go** yet again.
- After running the simulation once and allowing it to finish on its own, capture the image of both the **World** window and the **Disease Prevalence** plot in your simulation and paste them into a new document. Save this document with an appropriate name.
- Below the graphs that you just copied and pasted, record the duration of your outbreak. NetLogo records time in units that are called `ticks`. You can find the duration by typing the word `ticks` into the **Command Center**, in the box next to the prompt `observer>`. This number is also displayed on top of the **World** window, but rounded down to the nearest integer, and on the horizontal axis of the **Disease Prevalence** plot. Note that you may also hover your mouse over any part of this plot to see exact coordinates for any of the curves. This will show you the time and prevalence coordinates (in %) for a given point on the chosen curve.

Because disease transmission is based on random events (recall our discussion in Section 8.2.1), we do not know if our results are reproducible. We need to run a large number of simulations to make sure that the results are consistent with each other. Here is our plan:

- In the same document where you copied and pasted the panels from your first simulation, create a table similar to Table 8.1.
- To run a new simulation, click **New** and then **Go** again. Run 5 separate simulations by doing this 5 times. After each simulation, record in your table the total number of removed hosts, the duration of the outbreak (we can interpret 1 `tick` here as 1 week), and the maximum number of infectious hosts at a single point in time. You can find the latter value as the peak of the red curve in the **Disease Prevalence** plot or look it up in the window **Command Center** when you click the **Metrics** button after the simulation has terminated.
- Summarize your findings in two or three sentences. Are the data from your simulations similar? If not, explain how discrepancies may arise in separate runs of the simulation. How could you best summarize the overall result in terms of just one number for each variable? Should you just pick your favorite simulation? Or should you somehow combine the results from multiple simulations?

TABLE 8.1 Record of Differences in Simulations

Simulation #	Total # Removed	Duration	Max # Infectious
1			
2			
3			
4			
5			

As you may have correctly guessed, picking your favorite simulation rarely gives a reliable description of the epidemic. You may have wisely chosen to combine all of your data by taking the mean of each column to get the average number of removed hosts, average duration, and average maximum number of infected hosts. In addition to averages of these values, we may also be interested in knowing how much the data tend to differ from the mean. In other words, we may wish to report statistics such as the variance, standard deviation, maximum, minimum, or range of the data set. For this experiment let's record the mean and the standard deviation for our 5 sets of outputs by adding two rows with these values to our existing table.

Experiment 2: Investigating Population Size

Next let's look at our population size. We have run all simulations for a population size of 10. What do you think would happen if we increased the size of the population? Can you predict how this might change the mean and the standard deviation of the data sets?

Let's investigate your theories.

- Change the population size to 20 hosts by setting **num-nodes** to 20, but make sure to leave all other parameters the same. Click **New**. The first thing you may notice is that the **World** window now shows 20 nodes, where each node is still connected to every other node, producing quite a beautiful picture.
- Create a new table just like the previous one, only make sure to label this table appropriately so that you know it corresponds to data from a population of 20 instead of 10 hosts.
- Run 5 more simulations with this new population size (by clicking **New** and then **Go** each time), and remember to record the data in your new table after each simulation. You may wish to speed up the simulation by moving the speed slider to the right. This will not affect the results of your simulation; it will simply allow you to view the results more quickly.
- After all data have been entered into the table, calculate and record the mean and standard deviations as you did in the previous experiment.
- Now compare your results from when the population was 10 to when the population increased to 20. Record any differences you observe in how the population size appears to affect the mean and standard deviation, and provide reasoning as to why you believe these differences occurred.

We have now run two experiments, each with 5 simulations. How confident are you in these results? How do you think we could improve the confidence in our results?

You may have again correctly decided that running a larger number of simulations could provide an even better estimate of the true mean and standard deviation for each of these output variables. How many simulations do you think we would need? Well, it turns out that we do not have to merely guess or pick our favorite number. There is a theorem that can help us out—the Central Limit Theorem. This theorem allows us to determine the minimum number of simulations needed to be 95% confident that our observed mean is within a certain error tolerance from the true mean. A brief description of the theorem can be found in the review of probability theory on our website [3].

Experiment 3: Using the Central Limit Theorem

Our current question is now how many simulations to run. Let us look specifically at the output variable **Total # Removed**.

Exercise 8.19. Using your results from Experiment 1 and the Central Limit Theorem, calculate the number of simulations needed to obtain 95% confidence that the observed mean number of removed hosts is within an error tolerance of 0.5 from the true mean. □

Now we want to run this many simulations, but you may have already guessed this will take a while using our current method of manually clicking **New** and then **Go** each time. The good news is that there is a better way, and it is called "batch processing." NetLogo has built-in commands for running large batches of simulations and recording all of the outputs in one spreadsheet. The instructions below will walk you through the process.

- Reset **num-nodes** back to a value of 10.
- Open **Tools** → **BehaviorSpace**.
- Define a **New** experiment.
  - Enter a nice suggestive **Experiment name**.
  - In the **Repetitions** box, enter the number you derived by using the Central Limit Theorem above. For example, if you found that you needed at least 500 simulations, then just enter `500` here.
  - In the **Measure runs using these reporters** box, type the following, exactly as it appears below:
    ```
    count turtles with [removed?]
    ticks
    ```
  - Uncheck **Measure runs at every step**
  - In the **Setup commands** box, type
    ```
    new-network
    ```
  - Click **OK**.
  - Click **Run**.
  - A window will appear, asking how you want to save the data. Give instructions for saving the output. Check **Table output** and uncheck **Spreadsheet output**. Unless instructed otherwise, leave the number of **Simultaneous runs in parallel** at its default value. Then click **OK**.
  - Next select the directory where you want to save your output.
  - After you click **Save**, a window should appear that has checkboxes **Update view** and **Update plots and monitors.** To speed up the process, uncheck both and move the speed control slider at the top of this window to the extreme right.

When the window finally disappears, the experiment is completed. You can now view the results of all of your simulations by opening the file you just created.

- Compute the means of the data in the columns with the headers `count turtles with [removed?]` and `ticks`. These should be the last two columns in the spreadsheet. These values represent the observed mean of the number of removed hosts and the duration of the outbreak.
- Next compute the standard deviations of the data in both of these columns.

Summarize your results from this experiment, including a statement of how confident you are that your results are close to the true values of the mean number of removed hosts and the mean duration of the epidemic.

Remember to save all files created in this section in case you need them for future use.

8.3 SENSITIVITY ANALYSIS

In Section 8.2 you observed that even when all parameters remained fixed, the output variables were different for each simulation run due to the random events involved in disease transmission. In this section we investigate how changing parameters of the model affects the output variables. This is known as *sensitivity analysis*.

When analyzing a model, it is important to consider the model's sensitivity to assumptions and parameters included in the model. If the output of a model changes significantly in response to small changes in a parameter value, we say the model output is *sensitive* to that parameter. Such analysis plays a key role in both model validation as well as predictions obtained from the model. For instance, if it is determined that fluctuations in parameter A are causing the most variability in the final size of an outbreak, we could focus our efforts toward control measures that directly affect the value of that parameter. On the flip side, if fluctuations in parameter B cause little variation in the final size of the outbreak, control measures that influence parameter B are much less likely to be effective.

There are a wide array of sensitivity analysis methods, each method with its own strengths and weaknesses. Some methods work well for *deterministic* models, which are models where you get the same output for each simulation run if you keep all parameters fixed; other methods work better for stochastic models, where there is

variability in the outputs for different simulation runs even when all parameters are kept fixed. Some methods are only applicable to "linear" models, while others are better suited for "nonlinear" models.

In this section, we will limit our focus to the most basic type of sensitivity analysis, a *one-at-a-time (OAT)* approach. This will involve varying one parameter over a specified range of values while keeping all other parameter values fixed, and observing the resulting changes in the model outputs. For a thorough introduction to methods of sensitivity analysis, we recommend [5] or [6].

Getting Ready

Open the file IONTW, click **Defaults**, and change selected parameter settings as follows:

model-time → Continuous
infection-rate: 0.002
end-infection-rate: 0.075
num-nodes: 150
num/frac: 6
auto-set: On

It is best to save these settings once you have entered them, in case you want to complete this exercise in more than one session. To do so, simply select **File → Save As** from the toolbar, and save your file under an appropriate name.

The setting **num/frac** = 6 will generate initial states with exactly 6 infectious hosts. Press **New** and convince yourself that this actually works. The population size is now so big that you will no longer be able to visually distinguish individual lines between the dots in the **World** window.

Experiment 1: Testing Sensitivity to Changes in the Infection Rate β

In Section 8.2, you used the Central Limit Theorem to deduce the number of simulation runs needed to have a certain level of confidence that the observed mean of an output variable (such as **Total # Removed** or **Duration**) is within a certain error tolerance from the true mean. Typically you would use this result for the number of simulation runs needed *each* time you change the value of a specified parameter. However, for the purpose of this exercise and to assist in speed of simulations, let us all choose 100 as our number of simulation runs.

In this first experiment, we will test the effects of changing the values of the parameter β, which is controlled by **infection-rate** in IONTW on the outputs **Total # Removed** and **Duration**. Before we start the analysis, ask yourself, what would you expect to happen to each of these output variables if, for instance, we *increased* the value of β?

Now it is time to test your predictions! Before starting, make sure you have set all parameter values to the ones specified above. Our plan is as follows: we will use the batch processing feature of NetLogo to run our simulation 100 times for 5 different choices of the parameter **infection-rate**, for a total of 500 simulation runs. This process will be similar to how we ran batch processing in Section 8.2, with a slight twist. So let's get started.

- Create a new table as shown in Table 8.2.
- Open **Tools → BehaviorSpace**.
- Define a **New** experiment.
  - Enter a nice suggestive **Experiment name**.
  - In the **Vary variables as follows** box, scroll down to `["infection-rate" 0.002]` and change this to `["infection-rate" [0.001 0.0005 0.003]]`. Make sure you include the additional brackets here. This will tell NetLogo to start the parameter **infection-rate** at 0.001 and increment its value by 0.0005 until it reaches a final value of 0.003.

TABLE 8.2 Record of Differences in Simulations When Varying β by a Maximum of 50% Above and Below the Default of 0.002

	100 Runs	Total # Removed	Duration
$\beta = 0.001$	**Mean**		
	Standard Deviation		
$\beta = 0.0015$	**Mean**		
	Standard Deviation		
$\beta = 0.002$	**Mean**		
	Standard Deviation		
$\beta = 0.0025$	**Mean**		
	Standard Deviation		
$\beta = 0.003$	**Mean**		
	Standard Deviation		

- In the **Repetitions** box, enter `100`.
  The code will run 100 times for *each* of the 5 choices of the parameter **infection-rate** specified in the previous step.
- In the **Measure runs using these reporters** box, type the following, exactly as it appears below:
  ```
  count turtles with [removed?]
  ticks
  ```
- Uncheck **Measure runs at every step**.
- In the **Setup commands** box, type
  ```
  new-network
  ```
- After this step, your window should look something like Figure 8.3.
- Now proceed as in the experiments of Section 8.2:
  Click **OK**; next click **Run**.

  In the window that asks how you want to save the data, check **Table output** and uncheck **Spreadsheet output**. Then click **OK** and specify the directory where you want to save your output file.

  In the next window that appears, uncheck **Update view** and **Update plots and monitors** and move the speed control slider to the extreme right to speed up the process.

When the window finally disappears, the experiment is completed. You can now view the results of all of your simulations by opening the file you just created.

- One of the columns will contain the different parameter values for **infection-rate**, starting with 100 rows of the value 0.001, then 100 rows of the value 0.0015, and so on, for a total of 500 rows (i.e., 500 total simulations).
- In your spreadsheet, your sample number will appear in the column with heading `run number`. If you are running a machine with multiple processors, it is possible that some samples will be run in a different order. Therefore, before doing any analysis, sort your data by the run number (using the Sort command in your spreadsheet editor) to ensure the samples are in numerical order.
- Compute the mean and standard deviation of the first 100 rows of the data in the columns with the headers `count turtles with [removed?]` and `ticks`, which represent the total number of removed and outbreak duration, respectively. These should be the last two columns in the spreadsheet. Once you have sorted the

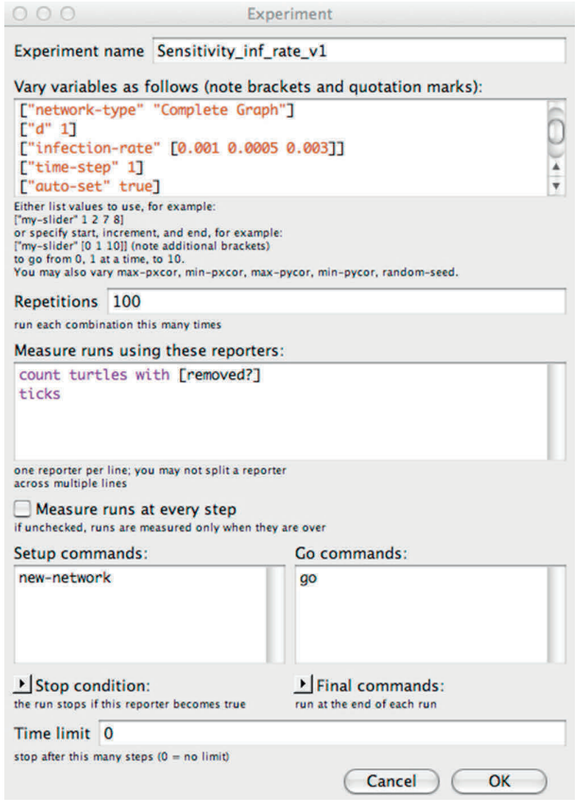

FIGURE 8.3 Sample batch processing window for sensitivity analysis.

data correctly, these values represent the observed mean number of removed hosts and the observed mean duration of the outbreak for **infection-rate** = 0.001.

- Next compute the mean and standard deviation of the values in the next 100 rows, where **infection-rate** = 0.0015. Continue this process until you have computed all different means and standard deviations for the five different values of **infection-rate**.
- Fill in all of your values into the table you created above.

What do you observe? How do changes in **infection-rate** affect the mean and standard deviation of **Total # Removed** and **Duration**? Is this what you expected to happen?

Experiment 2: Testing Sensitivity to Changes in the Number of Initially Infectious Hosts

Next let's test the model's sensitivity to the parameter **num/frac** that controls the number of initially infectious hosts. You will follow the same steps as in the previous experiment, only this time you will keep the value of **infection-rate** fixed at 0.002 and instead vary the value of **num/frac**. You may have noticed that in the previous experiment we varied **infection-rate** by a maximum of 50% above and below its default value of 0.002. Let us do the same for **num/frac**. First let us create a table as shown in Table 8.3.

- Follow the exact same steps as in the previous experiment, starting with the step "Open **Tools** → **BehaviorSpace**."
  - Use the same number of repetitions (100).
  - Make sure you reset the **infection-rate** parameter back to its default of 0.002 by typing `["infection-rate" 0.002]` in the appropriate line. If you set this parameter to `0.002` before opening the behavior space, it should already be set to this value automatically.

TABLE 8.3 Record of Differences in Simulations When Varying **num/frac** by a Maximum of 50% Above and Below the Default of 6

	100 Runs	Total # Removed	Duration
num/frac = 3	**Mean**		
	Standard Deviation		
num/frac = 4	**Mean**		
	Standard Deviation		
num/frac = 6	**Mean**		
	Standard Deviation		
num/frac = 8	**Mean**		
	Standard Deviation		
num/frac = 9	**Mean**		
	Standard Deviation		

  - Alter the values of **num-nodes** by entering `["num-nodes" 3 4 6 8 9]` in the **Vary variables as follows** box.
- Continue to follow all remaining steps as in the previous experiment. Remember to rename files appropriately to reflect this new experiment, so that you do not overwrite the data you collected in the first experiment.

After completing all of the steps, what do you observe? How do changes in the number of initially infectious hosts affect the mean and standard deviation of **Total # Removed** and **Duration**? Is this what you expected to happen?

Experiment 3: Testing Sensitivity to Changes in β and the Number of Initially Infectious Hosts for a Different Value of α

Now we are going to repeat the previous two experiments almost exactly, with just one small difference: we set **end-infection-rate** = 0.25. Recall that this parameter of IONTW controls the removal rate α.

Follow the exact same steps in Experiments 1 and 2, renaming saved files appropriately. Do you notice any differences now? What effects do you see when **infection-rate** is changed by a maximum of 50% above and below the value 0.002? What effects do you see when **num/frac** is changed by a maximum of 50% above and below the value 6? Do either of the output variables seem to be more sensitive to changes in one of these parameters? Can you describe how a specific output variable changes as you change each parameter?

Experiment 4: Testing Sensitivity to Changes in β and the Number of Initially Infectious Hosts for Yet Another Value of α

Now we are going to repeat Experiments 1 and 2 for **end-infection-rate** = 1.25. Follow the exact same steps in Experiments 1 and 2, renaming saved files appropriately. Do you notice any differences now? What effects do you see when **infection-rate** is changed by a maximum of 50% above and below the value 0.002? What effects do you see when **num/frac** is changed by a maximum of 50% above and below the value 6? Do either of the output variables seem to be more sensitive to changes in one of these parameters? Can you describe how a specific output variable changes as you change each parameter?

Concluding Thoughts

What conclusions might you draw from these experiments in terms of designing appropriate control measures? Paying close attention to your results in Experiments 3 and 4, what conclusions can you draw about this one-at-a-time sensitivity analysis approach? Does it seem reasonable to vary one parameter at a time while keeping all other parameters fixed? Why or why not?

Remember to save all files created in this section in case you need them for future use.

8.4 DETAILED INSTRUCTIONS FOR EXERCISE 8.14 OF SUBSECTION 8.2.5

In this exercise we will compare the predictions of continuous-time, discrete-time, and next-generation models with corresponding parameter choices. We will investigate how the final sizes and durations of outbreaks predicted by these models compare.

Getting Ready

Open the file IONTW, click **Defaults**, and change parameters to the following values:

model-time → **Continuous**
infection-rate: 0.01
end-infection-rate: 1
num-nodes: 150
num/frac: 10
auto-set: On

Experiment 1: Continuous-Time

In this experiment we will explore continuous-time models. For these models, the parameters are given as rates rather than probabilities. We will explain in Section 8.6 how, precisely, these rates translate into state transition probabilities. The rates have already been specified in the input fields **infection-rate** and **end-infection-rate** and they will be kept fixed throughout this exercise. We are ready to set up and run our first experiment. Define a **New** experiment with 500 repetitions. For all other settings and steps involved in running this and the other batch processing experiments in this exercise, follow the template that was described in detail in Section 8.2.

The experiment may take a while to run. Get yourself a cup of coffee. Text your friends. Call your mom.[1] When it is completed, you are ready to look at the output.

- Open the output file that you specified for your experiment.
- Compute the means of the columns (or rows, depending on the format you chose) with the headers `count turtles with [removed?]` and `ticks`.
- Divide the first number by the population size $N = 150$. This gives you the mean final size, as defined in Section 8.1 of the main text. The second number is the mean duration of the observed outbreaks.

The last two numbers give you a baseline that you will want to compare with the corresponding outputs of the next three experiments.

Experiment 2: Discrete-Time; $\Delta t = 0.02$

Discrete-time models can be treated as approximations to continuous-time models that are obtained by rounding up all state transition times to the nearest multiple of Δt. In the next two experiments we will explore how good the approximations are.

1. This message is brought to you by Correebugs, KeepShort, and Revizon, proud sponsors of excellence in publishing.

We will need to convert the rates of the continuous-time model into probabilities for the discrete-time models. These probabilities will depend on Δt. The button **Discrete Approx** of IONTW will do the conversion automatically; we will see in Section 8.6 how this calculation works. For this experiment, choose a very small time step $\Delta t = 0.02$. Change the previous parameter settings as follows:

model-time $\rightarrow$ **Discrete**

time-step: 0.02

Click **Discrete Approx** to automatically calculate **infection-prob** and **end-infection-prob**

Now you can define and run a **New** experiment with the same number of repetitions, output variables, and setup command as in the previous experiment. After the simulation is completed, analyze your output as before.

Experiment 3: Discrete-Time; $\Delta t = 1$

Now let us explore a discrete approximation to our continuous model with a large Δt. Let us choose $\Delta t = 1$. Change the previous parameter settings as follows:

time-step: 1

Click **Discrete Approx** to automatically calculate **infection-prob** and **end-infection-prob**

Now you can define and run a **New** experiment with the same number of repetitions, output variables, and setup command as in the previous experiment. After the simulation is completed, analyze your output as before.

Experiment 4: Next-Generation

Finally, let us explore the next-generation version of the model. Here we need to set $\Delta t = \langle \tau^I \rangle$ and **end-infection-prob** = 1.

First we need to figure out $\langle \tau^I \rangle$ for the continuous model as follows:

Set **model-time** $\rightarrow$ **Continuous**

Click **New** and then **Metrics**.

You can find the value of $\langle \tau^I \rangle$ for the continuous model by scrolling up the **Command Center** and finding `<tau>`.

Change the previous parameter settings as follows:

model-time $\rightarrow$ **Discrete**

time-step: [The value you found by looking up `<tau>`]

Click **Discrete Approx** to automatically calculate **infection-prob** and **end-infection-prob**.

Manually reset **end-infection-prob**: 1

Now you can define and run a **New** experiment with the same number of repetitions, output variables, and setup command as in the previous experiment. After the simulation is completed, analyze your output as before.

Analyze Your Data and Draw Conclusions

If the discrete and next-generation models are good approximations to the continuous-time model, then we should see very similar values for the mean final sizes and mean durations of the simulated outbreaks in all four experiments. Do we?

Rank the discrete models in terms of how well they appear to approximate the predictions of the continuous model. Which of the observed differences in the predictions of the four models with respect to final size or duration of an outbreak appear to be significant, and which may be due to random fluctuations?

We may ask you later in this chapter to take a second look at the data you collected from these experiments, so keep them in a safe place.

8.5 MODELING THE GENERATIONS OF AN INFECTION

In this chapter we have mostly focused on *whether* and *when* a given host i becomes infected. But in order to calculate probabilities for these events, we need to consider by *whom* host i might be infected at a given time. One can look at the problem from the opposite direction and consider whom a given host j might infect.

This point of view is exemplified by the second part of Project 8.1, where we ask you to study the example of an outbreak of measles in Australia. The outbreak was started by host $j^* = 1$ who introduced the virus into the Australian population. Host 1 infected hosts $2, 3, 4$, and 6, thus causing *4 secondary infections.* Host 6 in turn infected hosts 5 and 7, and host 7 infected hosts 8 and 9. Thus, hosts 6 and 7 caused two secondary infections each. Nobody else in Australia experienced infection during this outbreak.

In order to keep track of who infects whom, let us define the n^{th} *generation of the infection* as follows: Generation $Gen(0)$ comprises all initially infected hosts. In our example, $Gen(0) = \{1\}$. Generation $Gen(1)$ comprises all hosts that became infected by a host in generation $Gen(0)$, that is, by the index case. In our example, $Gen(1) = \{2, 3, 4, 6\}$. Generation $Gen(2)$ comprises all hosts that became infected by a host in generation $Gen(1)$. In our example, $Gen(2) = \{5, 7\}$. And so on.

Mathematicians have a clever way of writing "and so on." It is called a *recursive definition.* We would define $Gen(0)$ as above and then write: *Generation* $Gen(n + 1)$ *comprises all hosts that became infected by a host in generation* $Gen(n)$.

In terms of our example, the definition would tell us that if $n = 2$, then generation $Gen(2 + 1)$ comprises all hosts that became infected by a host in generation $Gen(2) = \{5, 7\}$ so that $Gen(3) = \{8, 9\}$. Generation $Gen(4) = Gen(3 + 1)$ must be the empty set $\emptyset$, as neither host 8 nor host 9 caused any secondary infections. If $Gen(n) = \emptyset$, then there are no hosts in $Gen(n)$ who could cause secondary infections, and the recursive definition implies that $Gen(n + 1)$ is also the empty set. Thus, in our example, $Gen(n) = \emptyset$ for all $n \geq 4$.

You can understand where the name "generation" comes from by thinking of each secondary infection as begetting a new infectious host, or you can take the point of view of the pathogens who first need to reproduce inside the host before their offspring can cause a secondary infection.

Now let us disregard the *physical times* at which secondary infections happen and think about how we would simulate the *next-generation process* by which secondary infections occur. We will reuse our terminology from Section 8.2.3 and assume that at step n of the process, host i is either in state I (which now means *included* in $G(n)$), state S (which now means *susceptible to inclusion* in $G(n + 1)$), or state R (*removed* and not available for inclusion in generation $G(n + 1)$). This looks very much like our states for *SIR*-models, but n stands for the generation number, not physical time.

In our example, the next-generation process would go through the following sequence of states:

$$(ISSSSSSSS\ldots), (RIIISISSS\ldots), (RRRRIRISS\ldots), (RRRRRRRII\ldots), \ldots \tag{8.5}$$

One could consider simulating a state sequence for the next-generation process by using the following pseudocode:

n0. Set n_{curr} to 0. Initialize the state of the population.
    Until instructed to stop **repeat**
n1. Generate the set *New* of hosts in the next generation of the infection.
n2. **If** $n_{\text{curr}} + 1 < T_{\max}$ **then** advance n_{curr} to $n_{\text{curr}} + 1$ **else** stop.
n3. **If** the current state of host i is I **then** change the state of host i to R.
n4. **If** $i \in New$ **then** change the state of host i to I.

Look familiar? This is exactly the same pseudocode that we used in simulating the time course of an outbreak in a next-generation *SIR*-model! Now you can see where the name that we gave to these models comes from.

But here we are talking about numbers n of generations instead of approximations t of physical time, so we changed the notation in the pseudocode accordingly. The first two lines of (8.5) may not represent the state sequence in the models that we investigated earlier in this chapter. These models and the next-generation process mirror different aspects of disease transmission. However, the above pseudocode shows that an agent-based

model that was designed for approximating the progression of an outbreak in physical time can be used for simulating the next-generation process.

Quite frequently, a mathematical construct (a model in our case) that was originally developed in one context can be used successfully in many other contexts. This is exactly what makes mathematical tools so powerful and versatile. Another example of course is probability theory, which was originally developed to study flipping coins and rolling dice in games of chance. It gives us powerful tools for understanding disease dynamics.

Many important aspects of disease dynamics can be expressed and studied in terms of generations, or even just in terms of the *numbers* $g(n)$ of hosts in generation $G(n)$. For example, take final size, the proportion of all hosts who will experience infection at *some* time during the outbreak: if host i experiences infection at some time, then host i must be included in *at least one* of the sets $G(n)$. Under the assumptions of *SIR*- or *SEIR*-models, host i can be included in *at most one* of these sets. Thus the sum of all numbers $g(n)$ gives us the total number of hosts that will experience infection at some time during the outbreak, and the final size can be expressed as

$$fsize = \frac{g(0) + g(1) + g(2) + \cdots + g(n) + \cdots}{N}. \tag{8.6}$$

You will already realize that the numbers $g(n)$ are random variables and cannot, in general, be predicted with certainty. What we *can* often calculate under suitable assumptions are their *expected values,* or *means* $\langle g(n) \rangle$. In particular, the *basic reproductive ratio* R_0 that will be introduced in Chapter 9 can be expressed in this terminology as follows:

$$R_0 = \langle g(1) \rangle. \tag{8.7}$$

8.6 HOW DO SIMULATIONS IN DISCRETE-TIME AND CONTINUOUS-TIME MODELS WORK?

In Section 8.3 of the main text we gave a detailed description of how simulations really work. The discussion was mostly restricted to next-generation models; here we will give details for more general discrete-time and for continuous-time models.

Let us first make a very important observation: as long as $t_{\text{curr}} < T_{\max}$, our simulator of Section 8.3.1 *determines the next state of the simulation exclusively based on information about the current state.* It is totally oblivious about how the system reached the current state and even about the current time, as long as it is less than $T_{\max}$. Our simulator has severe amnesia! We will see that this kind of amnesia makes our modeling a lot easier.

8.6.1 Continuous-Time Models

In a continuous-time model, the simulator needs to generate a time t_{next} at which the next transition event occurs, as well as the actual transition that happens at that time.

To see how this works, let us begin by considering the simplest case of an *SIR*-model with $N = 1$ host and assume that host 1 is infectious at time t_{curr}. Then the next transition event must be removal of host 1 at time $T_1^R = t_{\text{next}} > t_{\text{curr}}$. So our simulator only needs to determine t_{next}, or, equivalently, the value of the r.v. (random variable) $t_{\text{next}} - t_{\text{curr}}$. In terms of computer code, this boils down to drawing the value of $t_{\text{next}} - t_{\text{curr}}$ from a suitable probability distribution.

Now remember that our simulator suffers from amnesia. Thus $t_{\text{next}} - t_{\text{curr}}$ must be a *memoryless* continuous r.v. and hence must have an *exponential distribution* with probability density function $g(x) = \alpha e^{-\alpha x}$ for $x \geq 0$ and some parameter $\alpha > 0$ (see the review of probability theory at our website [3]). For this distribution, we will have

$$P(t_{\text{next}} - t_{\text{curr}} \leq \Delta t) = 1 - e^{-\alpha \Delta t} \approx \alpha \Delta t, \tag{8.8}$$

where the approximation on the right is valid as long as Δt is sufficiently small. In other words, the parameter α of the exponential distribution of $t_{\text{next}} - t_{\text{curr}}$ is the *rate* α at which infectious hosts are removed. This rate was already mentioned in Section 8.2.5 of the main text. In IONTW it is controlled by the input field **end-infection-rate**. This parameter does not give us t_{next} directly; it only determines the probability distribution from which t_{next} will be drawn.

Now let us consider the case of an *SI*-model with $N = 2$ hosts, and assume that at time t_{curr} host 1 is infectious while host 2 is still susceptible. Then the next transition event must occur during an effective contact between hosts 1 and 2 at time t_{next} so that $t_{\text{next}} = T_2^I > t_{\text{curr}}$. Again, our simulator only needs to determine t_{next} by drawing the value of the r.v. $t_{\text{next}} - t_{\text{curr}}$ from a suitable probability distribution.

As our simulator suffers from amnesia (remember?), $t_{\text{next}} - t_{\text{curr}}$ must be a memoryless continuous r.v. and hence must have an exponential distribution with some parameter $\beta > 0$. In analogy with (8.8), we will have

$$P(t_{\text{next}} - t_{\text{curr}} \leq \Delta t) = 1 - \mathrm{e}^{-\beta \Delta t} \approx \beta \Delta t, \tag{8.9}$$

where the approximation on the right is valid as long as Δt is sufficiently small. In other words, the parameter β of the exponential distribution of $t_{\text{next}} - t_{\text{curr}}$ is the *rate* β of effective contact between host 1 and host 2. This rate was briefly mentioned in Section 8.2.5 of the main text. In IONTW it is controlled by the input field **infection-rate**.

Now that we have examined the two simplest special cases, we are ready for the whole story. At time t_{curr} we may have some susceptible hosts, some infectious hosts, some removed hosts, and (if we consider an **E**-compartment) some exposed hosts.

The rates of removal may in general not be the same for all hosts. We let α_i denote the removal rate for host i. It follows from the properties of the exponential distribution that $\langle \tau_i^I \rangle = \frac{1}{\alpha_i}$. We use the notation α_i here to cover all possibilities. But in all agent-based models that we consider in this chapter and the follow-up Chapter 9, we will always have $\alpha_i = \alpha$ for a fixed constant α. This is the input value **end-infection-rate** in IONTW. The mean duration of infectiousness is then $\langle \tau^I \rangle = \frac{1}{\alpha}$. In the NetLogo model it can be computed by clicking **Metrics.** The output will appear in the window **Command Center** with the label `<tau>`.

Similarly, the rates $\beta_{i,j}$ at which effective contacts between hosts i and j occur may not be the same for all pairs (i, j) with $i \neq j$. In the agent-based models of this chapter, we always make the assumption that $\beta_{i,j} = \beta$ for some fixed constant β that represents the mean contact rate for all pairs of distinct hosts. This parameter is controlled by the input field **infection-rate**. But here we will use the notation $\beta_{i,j}$ to keep the discussion as general as possible. In Chapter 9 we will investigate some models where these rates $\beta_{i,j}$ may be different for different pairs of distinct hosts, but they will still be controlled by the input field **infection-rate** in conjunction with other model parameters.

In the context of models with an **E**-compartment, the rates $\beta_{i,j}$ represent rates of movement from the **S**-compartment into the **E**-compartment. The rate at which host i becomes infectious and moves from the **E**-compartment into the **I**-compartment will be denoted here by γ_i. It also represents the parameter of an exponential distribution. In our agent-based models we make the assumption that all $\gamma_i = \gamma$ for some fixed parameter γ. If **latent-period** is set to **On**, this parameter is controlled by the input field **end-latency-rate**.

Exercise 8.20. Using ideas similar to our discussion in the previous paragraphs, explain how our simulator would use these parameters in the case of an *SEIR*-model for a population of $N = 1$ hosts if at time t_{curr} host 1 is in the exposed state. □

A lot of parameters have been introduced. How does our simulator use them to generate t_{next} and the next transition event?

We have already seen how to do this when it is already known *what* will happen next and *to whom*, but not *when* the state transition will occur. In a real simulation, a number of different events could happen: an effective contact, onset of infectiousness of a previously exposed host, removal of an infectious host. How do we decide which one will occur next, when, and to whom?

Our simulator needs to make a choice based on an appropriate probability distribution. Conceptually, the easiest way of approaching this problem is to consider all possible scenarios:

- First, look at all infectious hosts i individually, pretend that nothing else is going to happen in the meantime, and then generate T_i^R as we did in our toy example with only one host.
- Next, look at all pairs (i,j) with i susceptible and j infectious at time t_{curr} and generate the time of the next effective contact between these two hosts.
- Finally, if there is an **E**-compartment, generate times T_i^I for all hosts who are in an exposed state at time t_{curr}, proceeding as in your solution for Exercise 8.20.

These steps would give our simulator a list of possible transition events and times when they *would* occur *if* no other transition had occurred in the meantime.

Exercise 8.21. The next transition event presumably should be one from this list. How could our simulator choose t_{next} and the particular state transition that will occur at this time from the list? □

This procedure, or *algorithm*, actually gives the correct distribution for drawing the next transition event. But for large populations the list might be a really long one and it would take too much computation time and memory space. The simulations of our NetLogo code use a neat shortcut that works much faster and gives the same distribution. It is called the *Gillespie algorithm* and was introduced in [7]. For a very readable description of the algorithm, see [8].

8.6.2 Discrete-Time Models

In the previous subsection, we did not mention any time units at all. Time could be measured in seconds, minutes, hours, days, and so on. The choice of unit would influence the actual numerical values of the parameters α and β of the continuous-time model, but would not affect the calculations in any other way. If we were to rewrite a model that has, for example, been written for time units that represent weeks and we were to rewrite it in terms of days, we simply would need to rescale these parameters accordingly. In order to not make potentially misleading suggestions and leave the interpretation open to the user, we did not include familiar time units in our IONTW interface or the instructions for the simulation exercises. The only time unit that we mentioned was the nonspecific `tick` that is internally used by IONTW.

Exercise 8.22. Suppose you have built a model and estimated parameters α and β for transition rates per day. How would you need to rescale these parameters if you were to reinterpret time in units of hours instead? Does this make sense in view of how the mean duration of infectiousness $\langle \tau^I \rangle$ is calculated from these parameters? □

The discrete-time models that were introduced in Section 8.2.5 of the main text assume that time proceeds in discrete steps, $t = 0, 1, 2, \ldots$. When these models are treated as approximations to continuous-time models, it does matter how time units are chosen. Each step t signifies that t units Δt of physical time have elapsed since the start of the simulation, where Δt is your chosen time unit.

For discrete-time models, the simulator no longer needs to randomly generate a time t_{next}, as we always will have $t_{\text{next}} = t_{\text{curr}} + 1$. Keep in mind, though, that this difference $t_{\text{next}} - t_{\text{curr}} = 1$ corresponds to Δt of physical time that is supposed to elapse between t_{curr} and t_{next}.

Now let us assume that we have a continuous-time model that is expressed in our favored time units, say days. Choose Δt that may correspond to a *different* time unit, say an hour. The code IONTW allows you to record this new time unit in the field **time-step** and then compute the corresponding parameters of the discrete-time approximation by clicking **Discrete Approx**.

Let us see how this works. For simplicity, we will ignore here the optional **E**-compartment.

Consider the simplest case of an *SIR*-model with $N = 1$ host and assume that host 1 is infectious at time $t_{\text{curr}} = 0$. Then the next transition event must be removal of host 1 at some time $T_1^R > 0$. In terms of the continuous-time model, T_1^R could be any positive real number. In the discrete-time approximation, we would *round up* this real number to the nearest multiple of Δt. There is a positive probability a that $T_1^R \leq 1$. This probability will be used in the discrete-time approximation as a parameter. It can be manually set by the input field **end-infection-prob** of IONTW. Alternatively, you can have the program calculate it automatically by clicking **Discrete Approx**. In view of how we defined discrete approximations in Section 8.2.5 of the main text,

the probability a will be equal to

$$a = P(T_1^R \leq \Delta t) = 1 - \mathrm{e}^{-\alpha \Delta t}. \tag{8.10}$$

Exercise 8.23.

(a) Suppose host 1 is infectious at time $t_{\text{curr}} > 1$ of the discrete model. How can you calculate the probability that host 1 will no longer be infectious at the next time step of this model?

(b) How would you calculate the probability $b_{i,j}$ of at least one effective contact between hosts i and j over a time interval of length Δt from the rate $\beta_{i,j}$ for an *SI*-model? □

In the agent-based models that we consider here, we need only a single parameter b. In this chapter it represents the mean of $b_{i,j}$ over all pairs of distinct hosts $i \neq j$; in Chapter 9 we will slightly modify this interpretation. By clicking **Discrete Approx**, you can compute b automatically in IONTW from the parameters of the continuous model and the value of **time-step**. Alternatively, you can enter this parameter manually in the input field **infection-prob**.

8.6.3 Comparing Continuous-Time, Discrete-Time, and Next-Generation Models

Continuous-time models appear to be closer to biological reality and one might expect that they would give us more realistic predictions than discrete-time approximations. On the other hand, simulations of discrete-time models sometimes run faster and discrete-time models are conceptually easier to understand. For this reason, we have developed our models mostly in the framework of discrete model time.

Let us now investigate when we would expect discrete approximations to continuous-time models to be reasonably accurate. Before you read on, please have another look at the data that you collected for Exercise 8.14 and compare them with our sample solution.

We found rather good agreement between the continuous-time and discrete-time models with a very small time step $\Delta t = 0.02$. This should come as no surprise. For intervals of such short length, it will be very rare that more than one transition event occurs within the same time interval. Moreover, when we translate the transition times of the discrete approximation back into physical time by multiplying by Δt, the rounded values will differ very little from the actual transition times of the continuous-time model.

For a large time step of $\Delta t = 1$ though, we observed significant differences between the continuous and discrete models, with the discrete approximation considerably overestimating the average final size and mean duration of outbreaks. The latter may be due to some extent to our rounding up in the construction of the discrete-time approximation, and to some extent to the fact that larger outbreaks tend to last longer. What really calls for an explanation are the observed differences in final size.

Let us revisit the solution for Exercise 8.23(b). If we consider an *SI*-model with $N = 2$ hosts and an initial state where host 1 is infectious and host 2 is susceptible, then T_2^I is an exponentially distributed r.v. with parameter β. From (8.10) we obtain the following value of the mean probability b of at least one effective contact between two distinct hosts $i \neq j$ over a time interval of length Δt:

$$b = P(T_2^I \leq \Delta t) = 1 - \mathrm{e}^{-\beta \Delta t}. \tag{8.11}$$

The calculations performed by our code after clicking **Discrete Approx** in fact are based on this formula. However, there is a snag: we derived the formula for an *SI*-model, where host 1 is guaranteed to stay infectious throughout the interval $(0, \Delta t]$. But in Exercise 8.14 we used the calculated value of b for simulating an *SIR*-model, where a host who is infectious at time t will be removed with probability a at some time *before* the end of the interval. If this occurs, then this host will be expected to cause on average *fewer* secondary infections than would be implied by the value of b that was calculated under the assumption of an *SI*-model. Thus our calculated parameters a, b would in many cases overestimate the expected numbers of secondary infection. This is exactly what we observed in the simulations with $\Delta t = 1$.

When a is very small, the distortions caused by this effect are insignificant, as in this case a simulation of a discrete model will rarely contain time steps when a host simultaneously causes a secondary infection

and is removed. This is what we observed in the simulations with $\Delta t = 0.02$, where $a = 0.0198$. However, for larger step sizes the distortions may become significant. In the simulations with $\Delta t = 1$, IONTW calculated $a = 0.6321$, which implies that about half of all effective contacts occur during the same interval where the infectious host involved in the contact gets removed. Sure enough, we saw a big difference in mean final sizes between the continuous-time model and its discrete approximation with $\Delta t = 1$.

There is a way to correct for this effect, and professional modelers would use the corrected version in the calculations that are triggered by clicking the **Discrete Approx** button. However, here our goal was not so much to give you a professional-grade tool, but to illustrate how approximations may distort the predictions of more elaborate models. Even the best possible approximations of more complex models by simpler ones are still approximations, and will almost always lead to *some* distortions, albeit hopefully minor ones.

The results for next-generation models fall in between the ones for the discrete approximations with $\Delta t = 0.02$ and $\Delta t = 1$. They are less accurate than the former, but considerably better than the latter. Let us try to understand why.

In these models we always use $\Delta t = \langle \tau^I \rangle$ and $a = 1$. This is in fact the way we set up next-generation models in IONTW: by setting **time-step** to 1 and **end-infection-prob** also to 1. They are discrete-time models, but not *discrete approximations* of continuous-time models. You can see this by considering (8.10): the right-hand side will always be less than 1 for any finite value of α.

Note that increasing the probability of removal by the next time step from $a = 0.6321$ in the discrete approximation with $\Delta t = 1$ to 1 will have the effect of shortening the duration of infectiousness for a proportion of $1 - a = 0.3679$ of all hosts who experience infection during an outbreak. Now we can see why the next-generation model predicts smaller final sizes than the discrete approximation.

Because the mean duration of infectiousness is the same in the continuous and the next-generation models, it remains still a bit of a mystery why the latter predicts larger average final sizes than the former. To understand the reason, we need to consider the function $a(\Delta t) = 1 - e^{-\alpha \Delta t}$ on the right-hand side of (8.10). Its first derivative $\frac{da}{d\Delta t} = \alpha e^{-\alpha \Delta t}$ is positive, and its second derivative $\frac{d^2 a}{d(\Delta t)^2} = -\alpha^2 e^{-\alpha \Delta t}$ is negative. Thus the function $a(\Delta t)$ is monotonically increasing and concave down. This means in particular that $a(0.5) > a(1) - a(0.5)$, which shows that hosts who do get removed until time 1 are significantly *more* likely to get removed during the first half of the interval $(0, 1]$ than during the second half of this interval. This has the effect of more than doubling the duration of infectiousness τ_i^I for a majority of hosts i for which $T_i^R - T_i^I \leq 1$ in the continuous model. On the other hand, next-generation models in effect shorten the interval of infectiousness of

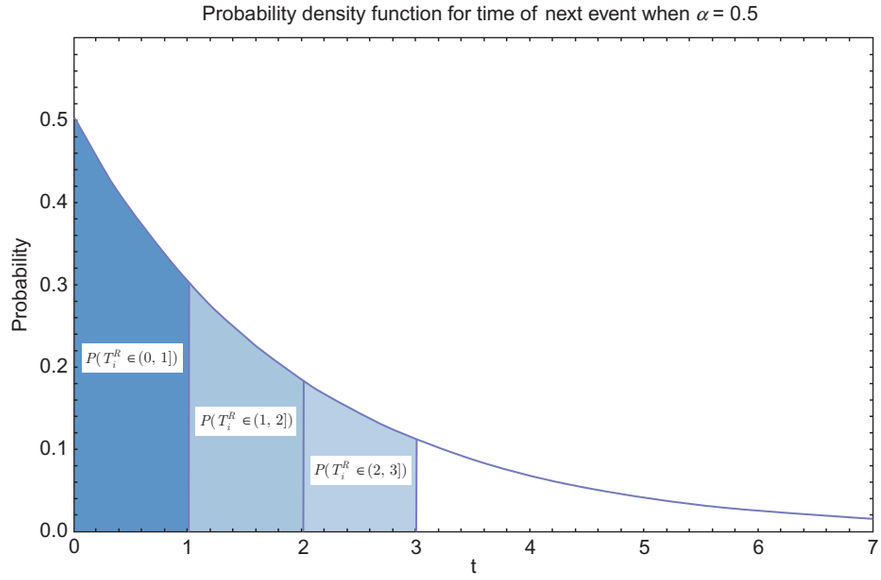

FIGURE 8.4 Probabilities of removal times during consecutive intervals for an exponential distribution.

hosts for which the duration of infectiousness is longer than 1 in the continuous-time model. However, for the parameter settings of Exercise 8.14, the first of these effects is stronger and results in an overestimate.

We started this subsection by indicating that continuous-time models appear to be closer to biological reality than discrete-time models are. In many respects this is true, but at least in one respect it is questionable.

For example, $\Delta t = \langle \tau^I \rangle = \frac{1}{\alpha} = 7$ days would be a reasonable parameter choice for a next-generation model of the seasonal flu. As Figure 8.4 illustrates, it follows from (8.10) that if $T_i^I = 0$, then our continuous-time models predict that

$$P(T_i^R \in (0,1]) > P(T_i^R \in (1,2]) > \cdots > P(T_i^R \in (6,7]) > P(T_i^R \in (7,8]) > \cdots \tag{8.12}$$

This is biologically unrealistic. Most people recover from the flu after 7 days, give or take a day or two. Actual distributions of durations of infectiousness tend to be normal rather than exponential. However, simulations that would use normal distributions of τ^I are much more cumbersome, beyond the reach of simulators with amnesia. There are various ways of getting around this problem. Using a discrete-time next-generation model is one of them. For diseases like the flu with strongly peaked distributions of τ^I, next-generation models may give more accurate predictions than continuous-time models based on exponential distributions. We should not automatically interpret the observed discrepancies between the two models as approximation errors or always give more credence to the continuous-time model.

8.7 SUGGESTIONS FOR FURTHER READING

For a more detailed introduction to basic probability than the short review on our website [3], we recommend [9].

Saltelli's text [5] provides a thorough and in-depth introduction to a variety of methods of sensitivity analysis, with a focus on global methods. Saltelli has also written several articles, each with a slightly different focus or application [10–12].

Students who want to learn more about mathematical models in epidemiology have a broad choice of introductory texts. The presentation given in [13] is easily accessible to readers with a limited mathematical background. Readers with solid background in elementary differential equations and linear algebra may find the books [14, 15] of interest. For upper-level undergraduate and beginning graduate students of mathematics, we highly recommend the survey article [16] that gives the most condensed treatment of mathematical aspects of compartment-level models and the textbook [17] that gives the most comprehensive and up-to-date treatment. The latter also contains many excellently structured exercises with hints and worked-out solutions.

8.8 PROJECTS

Project 8.1. This project illustrates some of the concepts covered in Section 8.1: the incidence and prevalence functions $nI(t)$, $|\mathbf{I}(t)|$, and a concrete example of contact tracing.

Suppose for a small outbreak of a disease the following numbers of cases were reported in each week:
Week 1: 6
Week 2: 18
Week 3: 36
Week 4: 19
Week 5: 7

If time is measured in weeks from the start of the outbreak, these data would give us the following estimate of the incidence function:

$$nI(1) = 6, \quad nI(2) = 18, \quad nI(3) = 36, \quad nI(4) = 19, \quad nI(5) = 7. \tag{8.13}$$

In Exercise 8.1, you already saw that one needs to be a bit cautious when making an inference from the number of case reports to the values of nI, but let us assume here that there is no underreporting and symptoms appear shortly after infection, so that the inference is warranted.

Now suppose that each person who started showing symptoms was hospitalized in a specialized health care facility for exactly 2 weeks, and that the release from the hospital coincided with cessation of infectiousness and symptoms.

Exercise 8.24. How many hospital beds were needed to cope with the outbreak? Express your answer in terms of the incidence and prevalence functions, and critically evaluate it.

For the second part of this project, we want you to read the article [18]. This article gives detailed information about a small outbreak of measles in Australia that affected 9 people. For this outbreak it was precisely determined who infected whom. You will learn from the article about some methods that are used in contact tracing. The information given in the article does allow you to give estimates of the times T_i^E of exposure for these 9 people. Read the article and all supporting materials carefully, and make a table of the best estimates of T_i^E that you can deduce from the article. Explain how you obtained the estimate for each host i. With what precision can you make this estimate for a given host based on the data? If these were the only data available on measles, what rough estimate could you derive about the mean duration $\langle \tau^E \rangle$ of latency and the mean duration of the incubation periods? When answering the last question, carefully consider which of the data points can be used for the estimate and whether they can give you information about the lower bound or the upper bound for one of these means. □

Project 8.2. This project is intended for students who want to gain some intuition about the meaning of "independence" in the context of the disease dynamics. It is assumed that students who attempt this project have worked through the material of Section 8.3.2 of the main text.

Consider a population that consists of Steve's friends. The only occasion on which these people ever meet are Steve's weekly wild Wednesday night (wwW) parties. And we mean *really wild*, where everybody hugs everybody else, everybody gets drunk, and people constantly mingle. Each of Steve's 99 friends attends about half of the wwW parties.

Now we need a little bit of mathematical notation. As you will see, the concept of independence is very subtle and we need to be very precise in our thinking. Consider a situation where Curr $= \{j_1, \ldots, j_9\}$ represents the subset of Steve's friends who attended the party this Wednesday and were at the time infectious with a highly contagious strain of the flu. For each $k = 1, \ldots, 9$, let F_k denote the event that Sally and j_k had an effective contact during the party, that is, a number of pathogens that would be sufficient to infect Sally got transferred from j_k to Sally. Are the events F_k independent?

Think about it this way: if i represents Sally and j_6 represents Ian, then the parameter $b_{i,6}$ of our agent-based models might represent your best estimate of the probability $P(F_6)$ in the absence of any further information. Of course, if we tell you more about what actually did or did not go on during the party between Sally and Ian, your estimate might change. But the editors requested that we remove any graphical details about the party from our draft. They only permitted us to tell you that Sally did *not* have an effective contact at this week's party with j_1, j_2, j_3, j_4, that is, with Pam, Patty, Peter, or Paul. Would this allow you to revise your estimate of $P(F_6)$ up or down from $b_{i,6}$? If effective contacts were like coin tossing, it would not, as the outcome for coin j_6 does not depend in any way on whether or not coins j_1, j_2, j_3, j_4 come up tails. But we told you that the party is really wild, and the flu strain highly contagious, so it seems that a rather unlikely event has happened.

Exercise 8.25. What would be the most plausible explanation? How does this relate to independence? □

Now assume that we had given you similar information, but had told the story in terms of Steve's quiet Sunday night round-table dinners, which Sally always attends, together with $j_1, \ldots, j_6$ and four of Steve's best friends.

Exercise 8.26. Redo the previous exercise for this new scenario. □

Oops, wait a minute. We just checked our records. It was at one of *last* year's wwW parties that Pam, Patty, Peter, and Paul were all infectious and Sally did not make effective contact with any of them. We apologize for the oversight.

Exercise 8.27. Should we revise our estimate $b_{i,6}$ of the probability that Sally made effective contact with Ian at *this* week's wwW party? □

REFERENCES

[1] Wilensky U. NetLogo home page. Available from: http://ccl.northwestern.edu/netlogo/ [accessed 2014.09.18].
[2] Wikipedia Entry. NetLogo. Available from: http://en.wikipedia.org/wiki/NetLogo [accessed 2014.09.18].
[3] Just W, Callender H, LaMar MD. Exploring transmission of infectious diseases on networks with NetLogo. Available from: https://qubeshub.org/iontw and also http://www.ohio.edu/people/just/IONTW/
[4] Just W, Callender H, LaMar MD. Online Appendix: Disease transmission dynamics on networks: network structure vs. disease dynamics. In: Robeva, R, editor. Algebraic and discrete mathematical methods for modern biology. New York: Academic Press; 2015. Available from: http://textbooks.elsevier.com/web/product_details.aspx?isbn=9780128012130.
[5] Saltelli A, Tarantola S, Campolongo F, Ratto M. Sensitivity analysis in practice: a guide to assessing scientific models. London: John Wiley & Sons; 2004.
[6] Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, et al. Global sensitivity analysis: the primer. Chichester: John Wiley & Sons; 2008.
[7] Gillespie DT. Exact stochastic simulation of coupled chemical reactions. J Phys Chem 1977;81:2340-61.
[8] Higham DJ. Modeling and simulating chemical reactions. SIAM Rev 2008;50:347-68.
[9] Tijms H. Understanding probability. Cambridge: Cambridge University Press; 2012.
[10] Saltelli A, Ratto M, Tarantola S, Campolongo F. Sensitivity analysis for chemical models. Chem Rev 2005;105:2811-28.
[11] Saltelli A. Sensitivity analysis for importance assessment. Risk Anal 2002;22:579-90.
[12] Saltelli A. Sensitivity analysis: could better methods be used? J Geophys Res Atmos (1984-2012) 1999;104:3789-93.
[13] Vynnycky E, White R. An introduction to infectious disease modelling. London: Oxford University Press; 2010.
[14] Brauer F, Castillo-Chavez C. Mathematical models for communicable diseases; 84. SIAM 2012.
[15] Keeling MJ, Rohani P. Modeling infectious diseases in humans and animals. Princeton, NJ: Princeton University Press; 2008.
[16] Hethcote HW. The mathematics of infectious diseases. SIAM Rev 2000;42:599-653.
[17] Diekmann O, Heesterbeek H, Britton T. Mathematical tools for understanding infectious disease dynamics. Princeton: Princeton University Press; 2012.
[18] Beard F, Franklin LJ, Donohue SD, Moran R, Lambert S, Maloney M, et al. Contact tracing of in-flight measles exposures: lessons from an outbreak investigation and case series, Australia, 2010. WPSAR J 2011;2:25-33.