Chapter 9

Online Appendix

Winfried Just[1], Hannah Callender[2] and M. Drew LaMar[3]
[1]Department of Mathematics, Ohio University, Athens, OH, USA, [2]University of Portland, Portland, OR, USA,
[3]The College of William and Mary, Williamsburg, VA, USA

In these modules, you will be working with our NetLogo program IONTW (Infections On NeTWorks). At the beginning of each unit, we will specify parameter settings that will be used for (most of) the unit. We restrict ourselves to listing the ones that are relevant to the work in the given part of the module. The exercises in each part build on each other and should not be done out of sequence. We will often specify how to change one or two parameters; the tacit assumption is that the other parameter settings will be as in the preceding exercise. If you need to interrupt work on one part and resume at a later time, save the current settings under an appropriate file name by using **File** → **Save As**.

TEMPLATE FOR BATCH PROCESSING

In our modules, we will frequently need to run large numbers of simulations in batch processing mode. There is a fixed sequence of steps that you need to follow to set up and run such an experiment. Only four of them change from experiment to experiment. Here we give you the template; in the text of the modules we will only indicate the ones that are marked here with asterisks and described in `typewriter font`. We strongly recommend that you keep all output files in a safe place in case you want to return to them later.

- Open **Tools** → **BehaviorSpace**
- ***** `Choose one of the following two options:`
  - Define a **New** experiment.
  - **Edit** an existing experiment.
- Enter a nice suggestive **Experiment name**. Make sure to choose a different name each time so as not to lose previously saved output files from prior experiments.
- **Repetitions:** ***** `Enter number of repetitions`
- **Measure runs using these reporters:**
  ***** `Enter the names of the desired output variables`
- Uncheck **Measure runs at every step**
- **Setup commands:**
  ***** `Enter the commands for initializing each network`
- Hit **OK**
- Hit **Run**
- Give instructions for how and where to save the output. Use the option **Table**, which gives cleaner 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.

Algebraic and Discrete Mathematical Methods for Modern Biology. http://dx.doi.org/10.1016/B978-0-12-801213-0.00029-0

- After you click **Save** the simulation starts running.
- A window should appear that has checkboxes **Update view** and **Update plots and monitors.** For best performance, uncheck both and move the speed slider in this window to the extreme right. When the window finally disappears, the experiment is completed.

9.1 MODULE 9.1: EXPLORING COMPARTMENT-LEVEL MODELS

9.1.1 Detailed Instructions for the Exercises in Section 9.2.2

Exercise 9.5

Open IONTW, click **Defaults**, and change the following settings if necessary:
model-time → Discrete
infection-prob: 0.09
end-infection-prob: 1
network-type → Complete Graph
gain-immunity: On
latent-period: Off
num-nodes: 10
auto-set: On
Click **New** and then **Metrics**. You can now find the value of R_0 for this model by scrolling up the **Command Center** and finding the value reported under `R0 =`.
Set the speed control slider to a slow speed for comfortable viewing and run 8 repetitions in slow motion. □

Exercise 9.6

Modify the settings that you used for Exercise 9.5 as follows:
num-nodes: 100
infection-prob: 0.005
As in the previous exercise, use **New** and **Metrics** to find R_0.
Set up and run a batch processing experiment by using our template with the following specifications:
Define a **New** experiment.
Repetitions: 100
Measure runs using these reporters:

```
count turtles with [removed?]
```

Setup commands:

```
new-network
```

When the experiment is completed, open the output file that you specified for your experiment.
Find the maximum and the mean value of the column with the header

```
count turtles with [removed?]
```

Retain your data for further analysis in the next exercise.

Exercise 9.7

Modify the parameter settings of Exercise 9.6 as follows:
infection-prob: 0.0025
num-nodes: 200
Look up R_0, then set up and run a batch processing experiment with these new parameter settings as in the previous exercise.
Next, modify the parameter settings as follows:
num-nodes: 400
infection-prob: 0.00125

Look up R_0, then set up and run a batch processing experiment with these new parameter settings as previously.

When the experiment is completed, open the output files that you specified for your two experiments of this exercise as well as the output file for the previous experiment.

In each file, sort the column with the header `count turtles with [removed?]` and find the maximum and the mean values of this column. Express these values both in terms of actual numbers of hosts who experienced infection and in terms of final size, which represents a fraction of the population.

Now you are ready to answer the question in the exercise.

Exercise 9.8

Modify the parameter settings that you used for the previous exercises as follows if need be:

infection-prob: 0.02

end-infection-prob: 1

num-nodes: 100

Look up R_0, record its value, and then set up and run a batch processing experiment for 100 repetitions with these new parameter settings.

When the experiment is completed, open the output file that you specified for this experiment. Sort the column with the header `count turtles with [removed?]` and find the maximum and the mean values of this column. Express these values both in terms of actual numbers of hosts who experienced infection and in terms of final size, which represents a fraction of the population.

Look at the sorted data in the last column. Do you observe a clearly distinguishable jump in the numbers of hosts who experienced infection during the simulated outbreak that might correspond to the difference between minor and major outbreaks? If so, record the mean and maximum numbers for all outbreaks that you would classify as "minor" and "major," according to this observation, as well as the proportion of simulations in which you observed a "minor" outbreak.

Retain your data for further analysis.

Now, modify the parameter settings as follows:

num-nodes: 200

infection-prob: 0.01

Repeat the steps for the previous data settings, including making a record of R_0 and preparing a summary of your output data.

Next modify the parameter settings as follows:

num-nodes: 200

infection-prob: 0.0075

Repeat the steps for the previous data settings, including making a record of R_0 and preparing a summary of your output data.

Finally, modify the parameter settings as follows:

num-nodes: 100

infection-prob: 0.015

Repeat the steps for the previous data settings, including making a record of R_0 and preparing a summary of your output data.

How do the values that you calculated appear to depend on R_0 and the population size N?

9.1.2 Projects on Compartment-Level Models

Project 9.1. In this project, we will explore the effect of various control measures such as vaccination, quarantine, culling, tightening border controls, and other types of behavior modification on the epidemic curves and the predicted final sizes of outbreaks.

Recall the example of the 2009 H1N1 "swine flu" outbreaks that we discussed in Section 8.1 of Chapter 8. For this infection, it has been estimated that R_0 was between 1.7 and 1.8 [1]. To be specific in our calculations,

let us assume here that $R_0 = 1.75$. This may seem rather small compared with the other examples that were mentioned after Definition 9.1. But $1.75 > 1$, and Theorem 9.3 predicts a positive probability $1 - z_\infty$ of major outbreaks. It can be shown that for continuous-time models, the probability for a single index case to cause a major outbreak is $1 - z_\infty \approx 0.57$. Moreover, if a major outbreak does occur, then its final size is predicted to be about 0.71. In terms of the US population, this would translate into more than 200 million people who would experience infection during a major outbreak!

The details of these calculations go beyond the scope of this chapter, but we want to emphasize that these numbers are obtained under the uniform mixing assumption and would apply if *no control measures whatsoever were taken.*

In Section 8.1 of Chapter 8, we mentioned that four types of control measures were considered and implemented by health authorities of various countries as possible means for preventing major outbreaks of the 2009 H1N1 infection: Tightened border controls, vaccination, behavior modifications, and quarantine of confirmed cases. Here we want to examine the likely effect of these control measures on a possible outbreak in a given country.

How could we use our models to investigate this problem? First we need to translate these possible control measures into the mathematical terminology of our model.

Let us begin with border control. This type of control measure will prevent some infectious or preinfectious hosts j from entering our country and becoming index cases j^*. But detection of symptoms is not easy, and it may not be possible at all to detect preinfectious hosts who will eventually become infectious. Thus, eventually some host j_1^* will cross the border and become an index case. With some luck, more precisely, with probability $z_\infty = 0.43$, index case j_1^* will cause only a minor outbreak. But eventually another infectious host j_2^* will cross the border, and the probability of this one causing a major outbreak will again be very close to 0.57. And so on. This is like repeatedly tossing a biased coin that comes up heads with probability 0.57: Eventually the coin will come up heads. In other words, eventually an index case j_k^* will cross the border and cause a major outbreak. Without any additional control measures, the final size of the outbreak is still predicted to be close to 0.71.

Tightened border controls may delay the onset of an outbreak in a given country and perhaps buy us some valuable time to prepare. But all by itself this control measure will not alter R_0 or the expected final size. The effect of border controls on the epidemic curve is like the one shown in the middle panel of Figure 8.2 of Chapter 8.

Now consider a policy of imposing quarantine on every host who shows symptoms. If host j is quarantined shortly after showing symptoms, this will shorten the time span during which host j can infect other hosts. Such a policy will not affect the length of time during which j is *sick,* but it will decrease the duration of infectiousness τ_j^I (compare with Figure 8.3 of Chapter 8). If such a policy is consistently enforced, it will in effect reduce the expected value $\langle \tau^I \rangle$. It follows from Eq. (9.4) that quarantine translates into a *reduction* of R_0 to a smaller value $R_0^- < R_0$. However, we cannot normally prevent all secondary infections by quarantine, as there will always be a small time lag between the onset of symptoms (which may roughly coincide with the onset of infectiousness) and the moment when j is quarantined. So quarantine will not reduce $\langle \tau^I \rangle$ to zero; it will only achieve a reduction of $\langle \tau^I \rangle$ to a smaller number $\langle \tau^I \rangle^- = r\langle \tau^I \rangle < \tau^I$.

Exercise 9.22. How large or small does the reduction factor r need to be so as to prevent all major outbreaks? What does this imply for a successful policy of imposing quarantine? □

There are various ways in which informed citizens can alter their behavior in response to the news of the likelihood of an outbreak. For example, they can refrain from some types of socializing such as attending leisure activities in crowded places like football stadiums or concert halls. Alternatively or in addition, informed citizens can modify their behavior during a given contact, for example, they can refrain from shaking hands, or use hand sanitizer more frequently.

Exercise 9.23.

(a) Explain how these types of behavior modification will influence the following quantities that we introduced in this chapter: b, c, v, R_0. What is their likely effect on the epidemic curve?

(b) Does adoption of such measures also make sense if they are not sufficient to prevent all major outbreaks? □

How about vaccination? This works a little differently than behavior modification or quarantine: It effectively *removes* the vaccinated host from the pool of susceptible hosts. We can think of it as moving the vaccinated hosts into the **R**-compartment. Usually it takes some time before the effect of a vaccination kicks in, so we really would want to apply vaccination sometime *before* the start of an outbreak. In mathematical terms, we would model outbreaks that result from introduction of (usually one) index case into populations where a fraction of hosts are susceptible and the remaining hosts are removed. In symbolic terms, this would translate into an initial state $(|\mathbf{S}(0)|, |\mathbf{I}(0)|, |\mathbf{R}(0)|)$ with $|\mathbf{R}(0)| > 0$.

Exercise 9.24. How could culling be translated into our mathematical framework? □

Usually it is not possible to vaccinate the whole population in time. Vaccine is costly, not enough may be available, and not all hosts may consent to vaccination. What is the likely effect on the outbreak if we vaccinate only $K < N$ hosts?

Consider an initial state $(|\mathbf{S}(0)|, |\mathbf{I}(0)|, |\mathbf{R}(0)|) = (N - K - 1, 1, K)$ of an *SIR*-model. Because the hosts in the **R**-compartment are *removed*, they play no role whatsoever in the model. We may as well consider a model where the population would consist of the remaining $N - K$ hosts, with an initial state $(N - K - 1, 1, 0)$. Presumably, the probability b of an effective contact between two given hosts over a unit time interval Δt and the mean duration of infectiousness $\langle \tau^I \rangle$ would remain the same. Let R_0 denote the basic reproductive ratio in the model with N initially susceptible hosts, and let R_0^- denote the value of the basic reproductive ratio in the model with $N - K$ initially susceptible hosts. By Eq. (9.4), if N is sufficiently large,

$$R_0 \approx \frac{b\langle \tau^I \rangle N}{\Delta t}, \qquad R_0^- \approx \frac{b\langle \tau^I \rangle (N - K)}{\Delta t}. \tag{9.12}$$

It follows that

$$R_0^- \approx R_0 \frac{N - K}{N} < R_0. \tag{9.13}$$

Thus, vaccination will result in a *reduction* of R_0—an effect similar to the one we predicted for behavior modifications and quarantine.

Let us consider what this means in terms of the real world. If we could vaccinate absolutely everybody, no outbreak whatsoever would take place. This would be best. If we vaccinate a *sufficiently large* proportion of hosts so that $R_0^- \leq 1$, then outbreaks may still occur, but will be minor. *Some* hosts may experience infection, but with probability p that is arbitrarily close to 1, their number will be limited by universal bounds $B(p)$ as in Theorem 9.2. This is not as good as totally preventing all outbreaks, but in a very large population this means that for each individual host the probability of actually contracting the disease is vanishingly small. Protection in this sense is conferred upon *each* host, even the ones that were not vaccinated. This phenomenon is called *herd immunity* in the literature. Achievement of herd immunity requires only that a sufficient fraction of the population gets vaccinated. The *herd immunity threshold,* denoted by HIT, is defined as the minimum proportion of the population that needs to be vaccinated to achieve herd immunity according to our model.

Exercise 9.25.

(a) Use Eq. (9.13) to derive a formula for HIT in terms of R_0.

(b) Find HIT for the 2009 H1N1 infection. □

The phenomenon of herd immunity is somewhat counterintuitive, as it protects many hosts who are *not* vaccinated through vaccination of *other* hosts. It also poses ethical problems in the case of human diseases: Suppose we actually *could* vaccinate all hosts, but this would be costly. *Should* we save money by immunizing only a sufficient fraction of hosts, or should we aim for immunizing as many hosts as possible?

Exercise 9.26. Outline the pros and cons of immunizing the whole population versus only the fraction needed for achieving herd immunity. Discuss possible obstacles to implementing your favored policy. Debate the issues with a partner or in a group of students. □

Now let us see how herd immunity works in simulated outbreaks.

Open IONTW, click **Defaults**, and define the following parameter settings:

model-time → Discrete

infection-prob: 0.01

end-infection-prob: 1
num-nodes: 200
auto-set: Off

According to these settings, we will be exploring an *SIR*-model with next-generation option for time, in a population of $N = 200$ hosts with one index case in an otherwise susceptible population. The value of R_0 will be very close to 2 in this model. We will use this rounded value $R_0 = 2$ to simplify our calculations.

You will notice that we already used these settings in Exercise 9.8, so you may want to review the data you collected for this exercise. Also, review your estimates of the probability z_∞ of minor outbreaks and the expected final size $r(\infty)$ that you derived for this model in Exercise 9.9. These estimates apply to a situation where no control measures whatsoever are implemented.

Exercise 9.27. Calculate the herd immunity threshold HIT for this model and derive a prediction about how many hosts need to be vaccinated to achieve herd immunity. □

Now let us see what happens if we vaccinate exactly the number K of hosts that you found in your solution of Exercise 9.27. Click **New** to create a network.

First we need to vaccinate K hosts prior to starting a simulation. We can do so by using the line that starts with `observer>` right below the **Command Center**. Enter in this line

```
ask n-of K turtles [become-removed]
```

You don't want to type `K` here, but instead enter the number you found in Exercise 9.27.

You should see K gray dots on the screen. The rest should still be green.

Now enter in the same line

```
ask n-of 1 turtles with [susceptible?] [become-infectious]
```

One of the green nodes will turn red. Next, start a simulation by clicking **Go**.

Do you observe a major or a minor outbreak among the $200 - K$ nodes that did not get vaccinated? You might be able to tell by inspecting the plot **Disease Prevalence.**

As you already know, one simulation is not enough to derive any meaningful conclusions, so you may want to run a few more by clicking first **Last** and then **Go**. Using **Last** will save you the trouble of entering the lines for vaccinating hosts prior to the simulated outbreaks.

Does vaccinating *exactly* a proportion of HIT hosts work as desired? It is really difficult to tell from the plots. We need to do more serious work and analyze a large batch of simulations with these settings. Define and run a batch processing experiment by following the template and using the specifications:
Define a **New** experiment.
Repetitions: 100
Measure runs using these reporters:

```
count turtles with [removed?]
```

Setup commands:

```
new-network
ask n-of K turtles [become-removed]
ask n-of 1 turtles with [susceptible?] [become-infectious]
```

Note that we now need to include the commands that you previously typed into the `observer>` line with the setup conditions.

Exercise 9.28. Sort and analyze the column of your output file that represents the number of hosts who experienced infection in each simulation. Calculate the corresponding proportions of unvaccinated hosts who experienced infection.

Do the results indicate that the vaccination prevented all major outbreaks among the $200 - K$ hosts who did *not* get vaccinated? *Hint*: Keep in mind that the numbers in this column always include all vaccinated hosts. □

To be on the safe side, would it perhaps be better to vaccinate a slightly larger proportion than HIT of the population? And what would happen if we vaccinate a slightly smaller proportion than HIT of the population?

Exercise 9.29.

(a) Redo Exercise 9.28 for vaccinating $1.2K$ and $0.8K$ hosts, respectively.

(b) Compare your results with those of Exercise 9.28 in terms of the mean and maximum proportions of unvaccinated hosts who experienced infection and in terms of the proportion of simulations in which two or fewer hosts experienced infection.

(c) Is the effect of increasing vaccination coverage from 0.8HIT to HIT the same as increasing vaccination coverage from HIT to 1.2HIT? Discuss implications for public health policy. □

Exercise 9.30. Suppose we have only enough vaccine available to achieve a 10% coverage. Would this have much of an effect in terms of protecting the remaining 90% of the population? Explain how you would approach this question with the help of simulations, and discuss your findings in terms of implications for public health policy. □

Project 9.2. In this project we will investigate differences and similarities in the predictions of *SEIR*- and *SIR*-models.

Let us examine the dynamics of two infectious diseases, measles and diphtheria, using both *SEIR*- and *SIR*-models. Measles is a highly contagious viral infection that affects the respiratory system, causing symptoms including fever, runny nose, cough, and a rash covering the entire body. Approximately 1 or 2 out of every 1000 children infected will die from the disease. The United States experienced a record number of measles cases in 2014, with 593 cases and 18 outbreaks reported from January 1 to August 1 alone. From the time the United States started collecting data on measles in 2000, the second highest number of cases for a single year was in 2011, with just over 200 reported cases [2].

Measles has been observed to have the following epidemiological parameters of primary interest for our modeling: R_0= 14-18; latent period = 6-9 days; and duration of infectiousness = 6-7 days [3].

Diphtheria is caused by a bacterial toxin and can be deadly. Although symptoms start like a cold, with a sore throat and fever, they develop into a thick blue or grayish green coating in the back of the nose and throat, inhibiting breathing and swallowing. The disease can also inhibit proper heart function and attack the nervous system. Diphtheria takes the lives of approximately 1 out of every 10 people who become infected. Before a vaccine was developed in the 1920s, diphtheria was one of the most common causes of death in children in the United States. It is now very rare [4].

Diphtheria has been observed to have the following epidemiological parameters of primary interest for our modeling efforts: R_0=6.6; latent period = 2-5 days; and duration of infectiousness = 14-21 days [3].

Exercise 9.31. Based on the parameter values above, for both measles and diphtheria calculate α and β from Eq. (9.4) of Section 9.2.2 and γ as we explained in Section 8.6 of [5]. Assume a population size of $N = 200$ and use as midpoints the parameters that are given only in the form of ranges. Round all answers to 4 decimal places. □

Before completing the remaining exercises from this project, check the sample solution for the exercise above to ensure you calculated the correct values.

Prior to running simulations, consider how the inclusion of an **E**-compartment might alter the expected dynamics for each of these two diseases. Would the inclusion of the **E**-compartment produce a different mean final size for either disease? Would the inclusion of the **E**-compartment significantly alter the shape of the prevalence curves for either disease? If so, how? If not, why not? Would the inclusion of the **E**-compartment alter the mean duration of outbreaks? If so, how? If not, why not?

Now it's time to test our predictions. Let's look at diphtheria first.

Open IONTW, click **Defaults**, and change the following parameter settings:

model-time → Continuous

infection-rate: [Use the β value you calculated above for diphtheria.]

end-infection-rate: [Use the α value you calculated above for diphtheria.]

end-latency-rate: [Use the γ value you calculated above for diphtheria.]

latent-period: On

num-nodes: 200

auto-set: On

According to these settings, we will be exploring an *SEIR*-model of diphtheria, with continuous time, in a population of N = 200 hosts, and with one index case in an otherwise susceptible population. Click **New** and then **Metrics** to find NetLogo's approximation of R_0. If you entered all the correct parameter values, each rounded to 4 decimal places, you should have R_0=6.40847. Note that this number is smaller than the value $R_0 = 6.6$ that you used in your calculations. The reason is that IONTW uses a more accurate formula for R_0 than the approximation (9.4). We will discuss this more accurate formula in Module 9.5.

Run 5 simulations, each time copying and pasting the **Disease Prevalence** plot into a spreadsheet saved under an appropriate name. You can easily copy the graph by right-clicking on it and selecting "Copy Image."

Now simulate an *SIR*-model for diphtheria by changing **latent-period** to **Off** and keeping all the other settings the same. Run the simulation 5 times, each time copying and pasting the **Disease Prevalence** plot into the same spreadsheet.

Are there differences that you notice? Does the type of model (*SIR* or *SEIR*) seem to affect the mean duration of the outbreak or the mean final size? Is this what you expected from your initial predictions?

Let's now do a more rigorous investigation using NetLogo's batch processing capabilities. Reset

latent-period: On

Set up and run a batch processing experiment with the current parameters settings of an *SEIR*-model for diphtheria. Follow the template and use the specifications:

Define a **New** experiment.

Repetitions: 200

Measure runs using these reporters:

```
count turtles with [removed?]
ticks
```

Setup commands:

```
new-network
```

Exercise 9.32. Open the output file and sort the data by the column labeled `count turtles with [removed?]`. Try to find a clear jump in the number of removed hosts during the simulations. Based on this jump, group your data into those corresponding to minor and major outbreaks. Record the maximum durations for all outbreaks that you would classify as minor and major, according to this observation, as well as the proportion of simulations in which you observed a minor outbreak. Calculate the mean duration and mean final size for the groups of minor and major outbreaks separately. □

Exercise 9.33. Repeat Exercise 9.32 for the case of an *SIR*-model for diphtheria. This can be done by first changing **latent-period** to **Off** before defining a new batch processing experiment. Alternatively, you can edit the previous experiments by setting `["latent-period" false]` in the **Experiment** window of **BehaviorSpace**. □

After calculating the mean duration and mean final size for major and minor outbreaks, in both cases of an *SEIR*- and an *SIR*-model for diphtheria, what do you observe? How do changes in model type affect mean duration of outbreak and mean final size? Is this what you expected to happen? Explain.

Now let's take a look at how an *SEIR*- and an *SIR*-model differ when we change our focus to measles.

Repeat the preliminary explorations with 5 test runs that we did for *SEIR*- and *SIR*-models of diphtheria for measles instead. Here you will need to change the values of **infection-rate**, **end-infection-rate**, and **end-latency-rate** to the values you found for measles in Exercise 9.31. The value for R_0 that you get after clicking **New** and **Metrics** should be 14.7363.

Are there differences that you notice just by looking at these graphs? Does the type of model (*SIR* or *SEIR*) seem to affect the mean duration of the outbreak or the mean final size for measles? Is this what you expected from your initial predictions? Let's check these predictions yet again, using the same steps as we did above for diphtheria.

Exercise 9.34. Repeat Exercises 9.32 and 9.33 for measles instead of diphtheria. □

After calculating the mean duration and mean final size for major and minor outbreaks, in both cases of an *SEIR*- and an *SIR*-model for measles, what do you observe? How do changes in model type affect mean duration of outbreak and mean final size in this disease? Is this what you expected to happen? How would you explain the similarities and differences between the *SEIR*- and *SIR*-models for the two diseases that you investigated in this project? □

9.2 MODULE 9.2: GRAPHS IN NETLOGO

9.2.1 Representations of Graphs in NETLOGO

Open IONTW, click **Defaults**, and make sure that

auto-set: Off

The menu **network-type** gives you many options for creating different types of graphs. In Chapter 8 and Module 9.1 we worked only with the first of them, but now let us explore some of the other options. Set

network-type → **Erdos-Renyi**
num-nodes: 10
lambda: 3

We will explain later in this chapter what this particular network type and the parameter **lambda** stand for; right now just let us have some fun.

After clicking **New**, a graph with 10 nodes appears in the **World** window. When you click **New** again, *another* graph appears in the window. And so on. Interestingly enough, the option **Erdos-Renyi** does not give you a fixed graph, but *random graphs* of a certain type. We will see later in this chapter how one can use random graphs for disease modeling.

Let us look at some smaller examples. Change

num-nodes: 5
lambda: 1

Click **New** a few times and look at the resulting graphs. Can you make out their connected components?

Click **New** a few more times until you get a graph that has three or four edges. The **World** window gives the visual representation of this graph. Let us see whether we can figure out the corresponding mathematical representation. The button **Labels** acts as a toggle switch that shows or hides the numbers of the nodes. When you closely look at the numbers, you will notice that their numbering starts with 0 instead of 1, but this is just a different mathematical convention from the one we are using. For the next three exercises, the vertex set of a graph G with N nodes will be $\{0, 1, \ldots, N-1\}$.

Exercise 9.35. Find the sets $V(G)$ and $E(G)$ for the graph that is currently in the **World** window. □

The solution of Exercise 9.35 gives you a mathematical representation of a graph G in the **World** window. We can use mathematical representations of graphs in .txt-files and then import them into the NETLOGO program. To see how this works, save the file `sample-network.txt` that comes with our IONTW package in the same directory as IONTW. Then click **Load** and choose this file from the directory. A strange-looking graph will appear in the **World** window. Click **Labels** (twice) to see how the nodes are numbered.

Exercise 9.36. Have you seen a graph like this before? *Hint*: Think about shifting the numbering of the nodes by one so that it ranges from 1 to 8 instead of from 0 to 7. □

Now let us open the file `sample-network.txt` in a plain text editor and inspect it. The first line tells IONTW that this file represents a network. The next line specifies the number of nodes. The remaining lines specify one edge each, as you can see by comparing these lines with the picture in the **World** window.

Exercise 9.37.

(a) Create another plain text file `triangle.txt` that will give you a picture like Figure 9.1 in the **World** window after you click **Load**.

(b) Is there a less tedious way of making this graph show up? □

9.2.2 Paths and Distances

Open IONTW, click **Defaults**, and change the following settings if necessary:
network-type → **Nearest-neighbor 2**
num-nodes: 5
d: 1
auto-set: Off
Click **New** and **Labels** to see how the nodes of the graph in the **World** window are numbered.

Exercise 9.38.
(a) Find all simple paths from node 1 to node 3 and the distance $d(1,3)$.
(b) Are there any paths of length 4 from node 1 to node 3? If so, how many? If not, why can there be no such paths?
(c) Are there any paths of length 3 from node 1 to node 3? If so, how many? If not, why can there be no such paths? □

Now set
num-nodes: 6
Click **New** and consider the graph in the **World** window.

Exercise 9.39.
(a) Find all simple paths from node 1 to node 4 and the distance $d(1,4)$.
(b) Change **d** from 1 to 2 and click **New**. What happens to the number of simple paths from node 1 to node 4 and to $d(1,4)$ if you add new edges in this way?
(c) Does the addition of new edges in point (b) affect the distances between *all* pairs of nodes? If not, for which pairs do the distances remain the same? □

9.2.3 Degrees of Nodes

Open IONTW, click **Defaults**, and change the following settings if necessary:
network-type → **Nearest-neighbor 2**
num-nodes: 6
d: 1
auto-set: Off
Click **New** and **Labels** to see how the nodes of the graph in the **World** window are numbered.

Exercise 9.40. Consider the graph in the **World** window.
(a) Find the degrees of nodes 1 and 3, as well as the mean degree $\langle k \rangle$ using Eq. (9.5).
(b) How do the results of point (a) change if you first increase **d** from 1 to 2 and then from 2 to 3? *Hint*: Sometimes edges overlap and you cannot clearly distinguish them in the **World** window.
(c) Are the graphs that you considered in this exercise regular, that is, k-regular for some k? □

Now change **network-type** → **Empty Graph** and click **New**. This will produce a very special graph with six nodes that is called—guess what?

Exercise 9.41.
(a) Find the degree of each node for the graph in the **World** window as well as the mean degree $\langle k \rangle$ using Eq. (9.5).
(b) Is this graph regular? □

The empty graph should not be confused with the *empty set*, which is equal to the set of edges of this graph. The empty graph itself is a different mathematical object. After all, when we clicked **New** we did not just get an empty screen!

Next, change the settings to
network-type → **Random Regular**
lambda: 2
Click **New** several times and see what happens. Each time you will see a different-looking graph, but always a 2-regular one.

Exercise 9.42. Are these graphs actually different as mathematical objects or just the same graph drawn in different ways? Explain your method for obtaining the answer. □

Now change the settings to
network-type → **Edos-Renyi**
num-nodes: 6
lambda: 2
Click **New** several times and observe the resulting graphs.

Exercise 9.43.
(a) Do you see (many) regular graphs?
(b) Does the mean degree $\langle k \rangle$ appear always to be the same whenever you click **New**?
(c) Closely examine the degrees in a couple of graphs that you create in this way. How are these degrees related to the information that is plotted in **Network Metrics**? □

For network type **Erdos-Renyi** the parameter **lambda** stands for the expected mean degree. This may not be very transparent from examining very small graphs, as there are rather large random fluctuations. Let's try larger networks of this type.

Exercise 9.44. Set **num-nodes** to 10 and **lambda** to 2 and click **New**.
(a) Calculate the mean degree $\langle k \rangle$ using Eq. (9.5).
(b) Set **num-nodes** to 20 and again calculate the mean degree $\langle k \rangle$. If this is getting tedious, try to find a shortcut that uses the info plotted in **Network Metrics**.
(c) Repeat part (b) by increasing **num-nodes** to 50. For your calculations, use the shortcut that you found in part (b).
(d) Does the mean degree appear to be reasonably close to **lambda** for large network sizes? □

9.2.4 Trees and Forests

Open IONTW, click **Defaults**, and make sure that **auto-set** is switched **Off**.

Look at the options in the menu **network-type**. Which of these options will give us trees, and which will not? You might think that the answer is obvious. But whenever we mathematicians are tempted to say "obvious," we should think twice.

Exercise 9.45. For each of the following parameter settings, create a network, and determine whether the network is a tree or forest. Record this information in a table, together with the numbers of nodes and edges for each of the graphs.
(a) **Empty Graph; num-nodes** = 1, 2
(b) **Nearest-neighbor 2; num-nodes** = 5, 6, 7; **d** = 1
(c) **Preferential Attachment; num-nodes** = 8; **lambda** = 1; **d** = 1
(d) **Random Regular; num-nodes** = 10; **lambda** = 1, 2 □

Obviously, if we had jumped to conclusions based on seeing the option **Regular Tree**, we would have overlooked a whole forest!

Exercise 9.46. Now look at the data that you collected for the previous exercise and form a conjecture on how the number of edges in a tree is related to the number of nodes. Try to confirm your conjecture for the tree G_T of Figure 9.3. Can you prove your conjecture? □

Usually, when we consider a tree, we want to single out one node i and call it the *root* of the tree. The set $\mathcal{N}_1(i)$ of all nodes that are adjacent to the root forms the first *level* of the tree. In general, for $\ell \geq 0$, level ℓ of the tree will be the set $\mathcal{N}_\ell(i)$ of all nodes that are at a distance of exactly ℓ from the root. Exercise 9.17 implies

that for $\ell \geq 0$, the set $\mathcal{N}_\ell$ consists of all nodes that are adjacent to some node in $\mathcal{N}_{\ell-1}$ except for the nodes in $\mathcal{N}_{\ell-2}$. The *height* of a tree T with root i is $h(T) = \max\{\ell : \mathcal{N}_\ell \neq \emptyset\}$. It is the largest distance of any node from the root.

We bet that you are itching by now to look at the trees that we would get when **network-type** is set to **Regular Tree**. In this option, the number of nodes is calculated automatically, and the parameter **lambda** controls the height $h(T)$ of the tree. The parameter **d** controls the degrees of the nodes. When you create a network with this option, set the speed control slider to the extreme right. You want to first click **New**, then **Spring**. After the tree has taken a nice shape, click **Spring** again. For larger values of **lambda** you may still need to wait a while until **Spring** has done a decent job. Get yourself a cup of coffee. Text your friends. Call your mom.[1] Then click **Scale** to make the picture fit the **World** window. The root always appears in the middle of the **World** window and is labeled 0.

Exercise 9.47. For each of the following parameter settings, create a regular tree and find the levels $\mathcal{N}_\ell(0)$ for $0 \leq \ell \leq h(T)$:

(a) **lambda** = 1; **d** = 8. This type of tree is called a *star tree.*

(b) **lambda** = 4; **d** = 2. This type of tree is called a *binary tree.*

(c) How, exactly, is the parameter **d** related to the degrees of the nodes? □

It is often convenient to analyze properties of a tree in terms of the levels.

Exercise 9.48. Show that the diameter $\mathrm{diam}(T)$ of a tree satisfies the inequalities

$$h(T) \leq \mathrm{diam}(T) \leq 2h(T). \tag{9.14}$$

Find sufficient and necessary conditions for these inequalities to be strict. □

9.3 MODULE 9.3: NETWORK-BASED MODELS OF DISEASE TRANSMISSION IN IONTW

Open IONTW, click **Defaults**, and change the following settings:

network-type → **Regular Tree**

lambda: 4

d: 2

Click **New** and get a nice display by following the steps that you learned in part 9.2.4 of Module 9.2.

Click **Set**. One node will turn red, which indicates that the node is now infectious. All edges between this node and adjacent nodes will be red, which means that an effective contact at the current time along these edges would be successful. There can be effective contacts between two green (susceptible) nodes, but they will not lead to disease transmission. A white edge indicates that no effective contact between its endpoints can be successful. The model does not permit any effective contacts between nodes that are not adjacent.

Let's play around with the initial state a bit before running a simulation. If you click **Reset** and then **Set**, usually *another node* will turn red. If you change **num/frac** below **set-state-to** to 2, then you can make 2 nodes initially infectious. Let's try a more interesting number than 2 and make 20 nodes initially infectious.

Exercise 9.49. Now some of the edges with red endpoints are red and some white. Does this make sense in terms of the interpretation of edge color that we gave above? □

Now change **set-state-to** → **Removed** while retaining all other parameter settings.

Click **Reset** and then **Set**. You will see some gray edges in addition to white ones. Essentially, we have created a subgraph of the original network by removing some nodes and all edges that connect to them. The green nodes and white edges form the subgraph of the original network that is *induced* by the green nodes.

Now let's learn how to control which particular nodes will be infectious or removed in the initial state.

Click **Reset**

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

Choose **set-state-to** → **Infectious**
Set **num/frac:** 1
Click **Labels**
How can we make sure that node 0 becomes the index case?
set-state-by → **Vector from input**
Click **Set**.
In the dialogue box that appears, type `[0]`. Be sure to include the square brackets.
Click **OK**
Did you succeed in making node 0 infectious? If you want to make nodes 7 and 11 initially infectious, you would need to follow the same steps, but enter `[7 11]` instead of `[0]` in the dialogue box.

Exercise 9.50. How could you use these capabilities to create an initial state in which nodes 0, 1, 2, 3 are infectious and nodes 4, 5, 6 are removed, while all other nodes are susceptible? □

OK, now we know how to control the initial state. Let us run some simulations. Make node 0 initially infectious while all other nodes are susceptible, and work with the following disease transmission parameters:
model-time → **Discrete**
time-step: 1
infection-prob: 1
end-infection-prob: 1
end-latency-prob: 1
gain-immunity: On
latency-period: On
Set the speed slider near the lower end of the "slower" range; adjust for comfortable viewing as needed. Click **Go**, sit back, relax, and enjoy the movie. Use **Last** and then **Go** to repeat a couple of times.

The yellow dots indicate that a host is currently exposed—it is not yet infectious, but will soon become infectious (red), not unlike traffic lights.

Exercise 9.51.
(a) Can you always figure out who infected whom?
(b) How is what you are seeing related to the notions of *path* and *distance?* □

Now let us change
end-latency: 0.5
Run a few simulations using **Last** and then **Go** to start them.

Exercise 9.52. How do the observations differ from the ones in the previous simulation? How would you explain the observed differences? □

Next change
infection-prob: 0.5
Click **Reset** and create a new initial state in which node 6 is the index case in an otherwise susceptible population. Run a simulation and observe what happens. Use **Last** and then **Go** to repeat a few times.

Exercise 9.53.
(a) How do the observations differ from the ones in the previous simulation? How would you explain the observed differences?
(b) The subgraphs that are induced by the green nodes may contain several connected components. In the original tree, there will always be a path between any two such components of the subgraph. What property in terms of disease transmission do all these paths have in common? □

Finally, change the following parameters:
infection-prob: 1
network-type → **Nearest-neighbor 2**
num-nodes: 13
d: 1

Click **New** and create an initial state with one infectious host in an otherwise susceptible population. Click **Go**, and enjoy the movie.

Exercise 9.54.

(a) Can you always figure out who infected whom?

(b) Change **num-nodes** to 14, and create a new network and initial condition with one index case. Simulate an outbreak and answer the same question as in part (a).

(b) Explain which property of the network was crucial for answering "yes" or "no" in parts (a) and (b). □

9.4 MODULE 9.4: NEAREST-NEIGHBOR NETWORKS IN IONTW

9.4.1 Nearest-Neighbor 1

Open IONTW and click **Defaults**. Use the following parameter settings:

network-type → **Nearest-neighbor 1**
num-nodes: 10
d: 2
auto-set: Off

Click **New** to create a network. The graph $G^1_{\mathrm{NN}}(10,2)$ will appear in the **World** window.

Now repeat for **d** = 1, 3, 4, 100. Then set **num-nodes** to 20 and repeat for the same values of **d**; finally repeat for **num-nodes** = 50. For large values of **num-nodes** you will no longer be able to visually distinguish all edges, but they are still there. Inspect the plots **Network Metrics** as you go.

Exercise 9.55.

(a) Are the graphs $G^1_{\mathrm{NN}}(N,d)$ regular? □

(b) How do the degrees of the nodes depend on the parameters **num-nodes** and **d**?

9.4.2 Nearest-Neighbor 2

Open IONTW and click **Defaults**. Use the following parameter settings:

network-type → **Nearest-neighbor 2**
num-nodes: 12
d: 1
auto-set: Off

Click **New** to create a network. The graph $G_{\mathrm{RR}}(3,4) = G^2_{\mathrm{NN}}(12,1)$ will appear in the **World** window. This is a regular grid with $m = 3$ rows and $n = 4$ columns, with a total of $N = mn = 12$ nodes.

Now create new networks with **num-nodes** = 13, 14, 16. In the first case you will get the rectangular grid $G_{\mathrm{RR}}(1,13) = G^2_{\mathrm{NN}}(13,1)$ (not much of a grid, actually); in the second case you will get the rectangular grid $G_{\mathrm{RR}}(2,7) = G^2_{\mathrm{NN}}(14,1)$ with $m = 2$ rows and $n = 7$ columns; and in the third case you will get the square grid $G_{\mathrm{RR}}(4,4) = G^2_{\mathrm{NN}}(16,1)$. Can you see how the input parameter **num-nodes** works?

IONTW always tries to make the grid as nearly square as possible for the specified number of nodes. Unfortunately, we cannot use IONTW to visualize and explore $G_{\mathrm{RR}}(m,n)$ for all pairs (m,n) of positive integers, but we will be able to study enough interesting grids to understand how diseases spread on such networks.

Exercise 9.56.

(a) For which of the following choices of m and n can we create the network $G_{\mathrm{RR}}(m,n)$ with the above software capabilities?

$$m = 2, n = 10; \quad m = 12, n = 14; \quad m = 7, n = 33.$$

(b) Find a necessary and sufficient condition on the pair (m,n) that allows for creation of $G_{\mathrm{RR}}(m,n)$ as a **Nearest-neighbor 2** network. □

Now let us see what happens if we set
num-nodes: 20
d: 2
This will give us the graph $G_{\text{RD}}(4,5) = G^2_{\text{NN}}(20,2)$. The parameter setting **d** = 2 puts diagonals into the grid.
Next, try **d** = 3, 4. The pictures becomes a little more crowded. IONTW will now include edges between nodes that are at a larger distance. Try to see how this works exactly by doing the following exercise. Be sure to compare your observations with our sample solution.

Exercise 9.57. For the purpose of this exercise, let (i,j) denote the node in row i and column j of the grid. For which pairs of nodes (i_1,j_1) and (i_2,j_2) will IONTW put an edge into $G^2_{\text{NN}}(N,d)$ when $d > 1$? *Hint*: Sometimes edges overlap and you cannot clearly distinguish them in the **World** window. □

9.4.3 Disease Transmission on Nearest-Neighbor Networks

Now let us see how diseases spread on nearest-neighbor networks. Let us first explore an *SI*-model on $G^1_{NN}(20,2)$.
Open IONTW, click **Defaults**, and use the following parameter settings:
model-time → **Discrete**
time-step: 1
infection-prob: 0.1
end-infection-prob: 0
gain-immunity: Off
network-type → **Nearest-neighbor 1**
num-nodes: 20
d: 3
auto-set: On
Set the speed slider near the lower end of the "slow" range; adjust for comfortable viewing as needed. Click **New** and **Go** and then **Go** again when nothing appears to happen any-more. Repeat for **d** = 2, 1 while keeping all other parameters constant.

Exercise 9.58.
(a) Does the parameter **d** appear to influence the final size? Does it appear to influence the speed at which the disease spreads? *Hint*: Because we need to manually terminate the simulation by clicking **Go** again, we need to deduce the speed from features of the prevalence curves.
(b) How would you explain the likely causes of the observed differences and similarities? □
Now change the following parameter settings:
network-type → **Random Regular**
num-nodes: 12
lambda: 1
Run a few simulations with 1 initially infectious node.
Then change **num/frac** to 2, run a few simulations, and then repeat with **num/frac** = 3.

Exercise 9.59. Based on your observations, form a conjecture about the set of hosts who will eventually experience infection. You may want to compare your answer with our sample solution before reading on. □
Let us see whether your conjecture of Exercise 9.59 holds up for another type of network. Change the following parameter settings:
network-type → **Erdos-Renyi**
num-nodes: 24
lambda: 1.5
Run a few simulations with 1, 2, and 3 initially infectious nodes. Does your conjecture appear to hold up?
Let's try something bigger. Set

num-nodes: 100
num/frac: 1
Create a few networks and try to visually discern the connected component of the index case right from the picture as it appears in the **World** window.

Exercise 9.60. Can you always clearly see this connected component? If not, how can you get a little help from IONTW with the visualization? □

Note that Exercise 9.60 involves an analog of what is known in the medical community as an off-label use of prescription drugs. The doctors prescribed our software for studying disease transmission, but it works really well for treating visual overload. Such off-label uses of IONTW can be really helpful for studying structural properties of graphs.

So far we have been working with *SI*-models, where **end-infection-prob** = 0. Now let's see what happens in *SIR*-models. Change the following parameter settings:
end-infection-prob: 0.05
gain-immunity: On
num-nodes: 24
Run 10 simulations with these settings. Then repeat with **num/frac** = 2.

Exercise 9.61. What do you observe? Does the conjecture that you formed in Exercise 9.59 hold up for *SIR*-models? If not, how would you modify it? Can you prove the modified conjecture? □

We will return to the solution of Exercise 9.61 shortly. But first let us look at some *SIR*-models on graphs $G^1_{\mathrm{NN}}(N, d)$. Change the following parameter settings:
infection-prob: 0.4
end-infection-prob: 0.05
network-type → **Nearest-neighbor 1**
d: 1
auto-set: Off
When you click **New** you will see what is called a *cyclic graph* with 24 nodes. Click **Labels** to see how the nodes are numbered.

Now let us see what happens if we vaccinate two nodes prior to an outbreak. Vaccinate nodes 0 and 10 and make node 4 the initially infectious node as you learned in Module 9.3.

Run 10 simulations from the initial state you created, using **Last** and then **Go** to initialize each simulation.

Exercise 9.62. What do you observe? How do these observations relate to the conjecture that you formed in Exercise 9.61? Do you perhaps want to revise your conjecture? □

We will return to the solution of Exercise 9.62 shortly, but first let us see whether we can make use of what you have learned so far for developing optimal vaccination strategies.

Change the following parameter settings:
infection-prob: 0.9
end-infection-prob: 1
d: 2
num-nodes: 120
Create an initial state where node 0 is the index case while nodes 10 and 100 have been vaccinated. Run a couple of simulations for this initial state. The most striking difference with the simulations for Exercise 9.62 that you will observe is this: Now all nodes tend to become infectious and the vaccinated nodes no longer act as barriers against the further spread of the infection.

Exercise 9.63. We changed four parameters relative to the previous exercise. Which one is responsible for this difference? □

The settings for Exercise 9.63 enforce a next-generation *SIR*-model on a contact network $G^1_{\mathrm{NN}}(N, 2)$ with $N = 120$ and $R_0 = 3.6$. A corresponding compartment-level model would predict that about 97% of outbreaks will be major, with an average final size of 0.9695, which translates to about 116 of the 120 hosts experiencing infection.

Set up and run a batch processing experiment with the current settings of simulations. Be sure to follow the steps in the template with the following specifications:
Define a **New** experiment.
Repetitions: 100
Measure runs using these reporters:

```
count turtles with [removed?]
```

Setup commands:

```
new-network
ask n-of 1 turtles [become-infectious]
```

The second line of the setup commands causes exactly one host to become infectious. We are using this command here instead of **auto-set**, which should be switched **Off** in order to set the stage for vaccinating some hosts in our next batch processing experiment.

Exercise 9.64. Analyze the column of your output file that represents the number of hosts who experienced infection in each simulation. Do the results confirm the predictions of the compartment-level model? Does the particular structure of this contact network appear to make it easier or more difficult for this particular disease to spread? □

For a disease with $R_0 = 3.6$, the herd immunity threshold is $\mathrm{HIT} = 1 - \frac{1}{R_0} = 0.7222$. Thus, the compartment-level model would predict that we need to immunize more than 72% of the population in order to achieve herd immunity. For $N = 120$ this translates into 86 hosts that need to be vaccinated. Let us see whether this still works for the network-based model.

Let us set up and run another batch processing experiment where we "vaccinate" 86 randomly chosen nodes by moving them to the **R**-compartment prior to the start of the simulated outbreak. Be sure to follow the steps in the batch processing template with the following specifications:
Edit the previous experiment as follows:
Enter a new **Experiment name** so as not to overwrite your previous output file.
Change **Setup commands:**

```
new-network
ask n-of 86 turtles[become-removed]
ask n-of 1 turtles with [susceptible?] [become-infectious]
```

The three lines for the **Setup commands** create both a new network for each simulation and an initial state with 86 randomly chosen vaccinated hosts and one index case, exactly in the order in which vaccinating part of a population prior to an outbreak would work.

Exercise 9.65. Analyze the column of your output file that represents the number of hosts who experienced infection in each simulation. Do the results confirm the predictions of the compartment-level model? *Hint*: Remember that now we have 86 hosts who are removed from the outset, and the disease can spread only among the remaining 34 hosts. □

Vaccine is costly and may be scarce. What if we had only enough of it available to vaccinate 10% of the population, that is, 12 hosts? The corresponding compartment-level model predicts for a disease with $R_0 = 3.6$ that vaccinating such a small proportion would make scarcely a dent; major outbreaks would presumably occur with probability ≈ 0.96, and on average about 96% of the hosts who were not vaccinated would experience infection.

Let us see what happens in our model that is based on the contact network $G^1_{\mathrm{NN}}(120, 2)$. Set up and run another batch processing experiment as follows:

Edit the previous experiment and give it a different **Experiment name**.
Change the second line of **Setup commands:**

```
ask n-of 12 turtles [become-removed]
```

Exercise 9.66. Sort the column of your output file that represents the number of hosts who experienced infection in each simulation and analyze the observed outcomes. Do the results confirm the predictions of the compartment-level model? □

Compartment-level models are based on the uniform mixing assumption and make no distinction between individual hosts. Thus with compartment-level models we cannot study any other vaccination strategy than vaccinating randomly chosen hosts. In network-based models though, we might consider targeting a particular set of hosts with vaccinations. It will be convenient for the next exercise to create a file in plain text format (with extension .txt) that specifies which nodes should be vaccinated. If you want to vaccinate hosts 1, 2, and 80, your file should look like this:

```
[1 2 80]
```

Now create such a file. Save it under the name `vaccinate.txt` in the same directory in which you are running IONTW, and then click **New**.

Enter the following in the `observer>` line:

```
ask turtles-from-file "vaccinate.txt" [become-removed]
```

Click **Labels** to check whether the result is as you had expected. Once everything works fine, you can use the line

```
ask turtles-from-file "vaccinate.txt" [become-removed]
```

in the setup commands for batch processing instead of

```
ask n-of 12 turtles [become-resistant]
```

Exercise 9.67. How would you target vaccination to a particular set of 12 hosts so as to achieve the best possible result for the available amount of vaccine? First try to think of the best way to choose the hosts that will be vaccinated. Then run a batch of simulations similar to the ones of Exercise 9.66 for this choice of vaccinated hosts. Compare the results with the ones of the previous exercise in terms of minimum, maximum, and mean final sizes of the outbreak. □

Now let us return to your solution for Exercises 9.59, 9.61 and 9.62. Your conjecture might have been a version of the following theorem.

Theorem 9.4. *Consider any network-based model of disease transmission as defined in Section 9.3.2 of the main text; let G denote the underlying contact network, and let G^- be the subgraph that is induced by the nodes that are* not *already removed in the initial state. Assume that there is no loss of immunity so that removal is permanent. Then the model predicts the following:*

(a) *During any given outbreak, host i will experience infection* only if *the connected component of i in G^- contains an initially infectious node j*.*

(b) *Assume furthermore that the model is of type SI or of type SEI. Then, during any given outbreak, host i* will *experience infection* if *the connected component of i in G contains an initially infectious node j*.*

Theorem 9.4 is very general. It applies to both continuous-time and discrete-time models. Moreover, part (a) applies to models of types *SI*, *SEI*, *SEIR*, *SIR*, and *SIS*. In contrast, point (b) applies only to *SI*- and *SEI*-models. There is an important difference in terms of what the theorem predicts for the latter model types as opposed to models of other types.

In terms of real-world applications, mathematical theorems allow us to predict and control the behavior of systems that reasonably well conform to the assumptions of the model. Exercise 9.67 illustrates such an application. Moreover, the *proofs* of mathematical theorems give us insights as to *why* a system behaves in the way that the theorem predicts.

Exercise 9.68. Explain why Theorem 9.4 is true. Depending on your level of mathematical preparation, give either an informal argument or a formal mathematical proof. □

9.5 MODULE 9.5: NETWORK AND DISEASE TRANSMISSION PARAMETERS

9.5.1 Degree Distributions in Grids $G^2_{\text{NN}}(N, d)$

Open IONTW, click **Defaults**, and set the speed control slider to the extreme right. Use the following parameter settings:

network-type → **Nearest-neighbor 2**
num-nodes: 12
d: 1
auto-set: Off

Create a network by clicking **New**. Take a screenshot of the degree distribution in the plot **Network Metrics**. Find the value of the mean degree for this network by clicking **Metrics** and looking it up in the **Command Center**. Record this information.

Exercise 9.69.

(a) Repeat for **num-nodes** = 15, 21, 33.
(b) Conjecture formulas for Q_k, q_k, and $\langle k \rangle$ for rectangular grids $G_{\text{RR}}(3, n)$.
(c) Prove your conjectures of point (b).
(d) Find $\lim_{n\to\infty} \langle k \rangle$ for the networks $G_{\text{RR}}(3, n)$. □

Once you get the hang of Exercise 9.69, feel free to generalize to grids $G_{\text{RR}}(m, n)$ and $G_{\text{RD}}(m, n)$. But this is somewhat tedious and not particularly enlightening. The following is more interesting.

Exercise 9.70.

(a) Explore the behavior of $\langle k \rangle$ for $G^2_{\text{NN}}(N^2, d)$ when $d = 1$ is fixed and $N \to \infty$. *Hint*: Don't try exploring networks of size larger than 1000, as **Metrics** will take too long to complete its calculations.
(b) Conjecture a formula for $\lim_{N\to\infty} \langle k \rangle$ for the networks $G^2_{\text{NN}}(N^2, 1)$.
(c) Prove your conjecture of part (b).
(d) Do the results that you observed in part (a) closely match your result for part (c)? □

Exercise 9.71. Mathematicians often talk about *finite size effects*. In our context, this means that theoretical results that can be proved for large network sizes N do not show up clearly if N is small or moderate. Did you observe this phenomenon in your work on Exercises 9.69 and 9.70? If so, in which direction did finite-size effects distort the theoretical predictions that you derived for $N \to \infty$? □

9.5.2 Exploring Neighborhoods with IONTW

Open IONTW, click **Defaults**, and use the following parameter settings:

model-time → **Discrete**
time-step: 1
infection-prob: 1
end-infection-prob: 0
end-latency-prob: 1
latent-period: On
network-type → **Nearest-neighbor 1**
num-nodes: 20
d: 1
auto-set: On

The network settings will give you the graph $G^1_{\mathrm{NN}}(20, 1)$. Create a network and an initial state with exactly one infectious node by clicking **New**.

The parameters for the spread of the disease enforce a next-generation *SEIR*-model. More precisely, because **end-infection-prob** is set to 0, infectious hosts will never recover. The model is actually of type *SEI*. What you will see during the simulation of an outbreak is green (susceptible) nodes turning yellow (exposed), and then to red (infectious), with no nodes turning gray (removed). Because we are simulating a discrete-time model, these changes will take place in a sequence of distinct steps. For **time-step** = 1, each step corresponds to one `tick` in the NetLogo language.

Exercise 9.72.

(a) Click **Labels** to see how the nodes are numbered. Before starting the simulation, write out the neighborhoods $\mathcal{N}_\ell(j^*)$ and $\mathcal{N}_{\leq \ell}(j^*)$, where j^* is the initially infectious node and $\ell = 1, 2$. Keep these results in mind when you observe the simulation.

(b) Set the speed control slider to about one-third of the slower range and adjust for comfortable viewing if needed. Click **Go** and observe what happens. Because this is an *SEI*-model, you will need to terminate the simulation by clicking **Go** again.

(c) Explain how the color changes at each step are related to the neighborhoods that you found in part (a). □

Now set

network-type → **Erdos-Renyi**

lambda: 2

Exercise 9.73. Repeat Exercise 9.72(b) for this model. Perform five runs. What is the most salient difference between the observations that you made for these networks and the observations you made in the previous exercise? How are these observations related to Exercise 9.21? □

9.5.3 Disease Transmission Parameters

Here we will closely examine the meaning of the disease transmission parameters. Let us start with the simplest kind of networks.

Open IONTW, click **Defaults**, and use the following parameter settings:

model-time → **Discrete**

end-infection-prob: 1

time-step: 1

network-type → **Empty Graph**

num-nodes: 128

num/frac: 128

auto-set: On

Click **New** to create a network and an initial state where all nodes are infectious.

No disease transmission can occur when the contact network has no edges. So the only disease transmission parameter of interest here is the probability a that a given infectious host gets removed after one time step. In IONTW, this parameter is controlled by **end-infection-prob**. If $a = 1$, then we should expect that all hosts will be removed after one time step, which corresponds to one `tick` if **time-step** = 1.

Let us see whether this is what happens. After clicking **Go**, all nodes should turn gray and the upper bar of the **World** window should show that the simulation has terminated after one `tick`. Does this work as expected?

Now let us set

end-infection-prob: 0.5

For $a = 0.5$, we should expect that at each time step about half of all remaining infectious hosts will be removed. Set the speed control slider near the lower end of the slow range; adjust for comfortable viewing as needed.

Use **Last** to make all nodes infectious, then watch what happens. Does it appear that at each step about half of all infectious nodes get removed? Because $128 = 2^7$, we should expect that it takes about 7 or 8 time steps until all nodes get removed, but due to stochastic fluctuations this may occasionally take more or fewer time steps. Do a few runs with fast speed to see whether this prediction holds up.

Now change the following settings:

model-time → Continuous

end-infection-rate: 0.6931

Recall that **end-infection-rate** controls the parameter α. Equation (8.11) of Section 8.6 of [5] gives the connection between this parameter and the parameter a of discrete-time models. Click **Discrete Approx** and convince yourself that our setting for α corresponds almost exactly to a value of $a = 0.5$ for discrete-time models.

Now click **Last**, move the speed control slider to a very slow setting, and start a simulation. You will see that nodes now turn grey one by one. Speed up the simulation, and after it terminates, look at the **Disease Prevalence** plot.

You will see that the red curve is decreasing and concave up. Thus, many more transitions from the **I**-compartment into the **R**-compartment occur near the beginning of the simulation rather than near the end. This is very similar to what we observed in discrete-time simulations. Moreover, the number on the `time` axis of the plot gives you the time when the last removal occurred in the simulation. It is no longer an integer, but should still on average be close to 7 or 8. Run a few simulations to check whether this prediction appears to hold up.

Next, let us explore the parameter b. In IONTW, this parameter is controlled by the input field **infection-prob**.

Change the following parameter settings:

model-time → Discrete

infection-prob: 0.5

end-infection-prob: 1

network-type → Regular Tree

lambda: 1

d: 10

auto-set: Off

Move the speed control slider to the extreme right and click **New** to create a star tree. Use **Spring** (twice) and **Scale** to make it look nice.

Make node 0, the root of the tree, initially infectious as you learned in Module 9.3. This will turn all edges into red ones, which means that an effective contact between any of the susceptible nodes and the root during the first step of the simulation will lead to a successful transmission. For any of the red edges, the probability that this will happen is governed by the parameter b, which is specified here by setting **infection-prob** to 0.5.

Exercise 9.74.

(a) How many secondary infections would you *expect* the index case to cause?

(b) Use **Last** to run about 10 simulations with this initial state, record the number of secondary infections for each simulation, and compute the mean. Do your results roughly conform with the expectation you formed in point (a)?

□

Wait a minute …"average number of secondary cases caused by one index case in an otherwise susceptible population" …we have seen this phrase before. Is this the same as R_0?

No. The definition of R_0 contained the phrase "average index case." Under the uniform mixing assumption, every node is average, but in a star tree node 0 is very special indeed.

Exercise 9.75.

(a) How would you need to modify your explorations to get a correct rough estimate for the model that we are currently exploring?

(b) Perform 20 runs of the relevant simulations and compare your results with those of Exercise 9.74. □

Now let us derive a tentative formula for R_0 in network-based models. Assume host j^* is the index case in an otherwise susceptible population. Host j^* can infect host i only if $\{j^*, i\}$ is an edge in the network. Let p be

the probability that there will be an effective contact between hosts j^* and i during the interval of infectiousness of host j^*. This probability will always be the same for all pairs (j^*, i) such that $\{j^*, i\}$ is an edge. Now we can think of the total number of secondary infections caused by host j^* as being obtained by tossing biased coins that come up heads with probability p. For each edge $\{j^*, i\}$, one coin is tossed. Its mean value will be the expected number of heads that come up in these tosses, which is equal to pk_{j^*} (see the review of probability theory on our website [6]), where k_{j^*} denotes the degree of node j^*.

In our current settings with $a = 1$, host j^* gets removed after exactly one time step, so that $p = b = 0.5$. Let's see how this works out if the index case is the root of a large star tree. Set

d = 100

Create a plain text file with the single line

```
[0]
```

and save it as `index0.txt` in the same directory where you keep IONTW. Set up and run a batch processing experiment for the current parameter settings by following the template with the following specifications:

Define a **New** experiment.

Repetitions: 100

Measure runs using these reporters:

```
count turtles with [removed?]
```

Setup commands:

```
new-network
ask turtles-from-file "index0.txt" [become-infectious]
```

When giving instructions for how and where to save the output, set

Simultaneous runs in parallel to 1 to avoid complications.

Exercise 9.76.

(a) Before looking at the output, ask yourself what you would expect in terms of the mean number of removed hosts at the end and the variability between runs.

(b) Now look at your output. Sort by the column that represents your output variable. Find the minimum, maximum, and mean of the column.

(c) Do your data match the expectations that you articulated in point (a)? □

So far, we have focused on one chosen index case. But we want to calculate R_0. Because in Section 9.2.2 of the main text this number was defined in terms of an "average" index case, we need to compute the mean over all possible choices of j^*:

$$R_0 = \frac{1}{N}\sum_{j^*=1}^{N} pk_{j^*} = p\frac{\sum_{j^*=1}^{N} k_{j^*}}{N} = p\langle k\rangle. \tag{9.15}$$

Great! Now we know, or so it seems. We still need to get a value for p. In our next-generation model that we used for Exercise 9.76 we had $p = b$, but in general things are no longer as simple as that.

Let us try a continuous-time model for a change. Change the following parameter settings:

model-time → Continuous

infection-prob: 0.6

end-infection-prob: 0.6

Exercise 9.77.

(a) Set up and run a batch processing experiment as for the previous exercise. What would you expect in terms of the output column?

(b) Sort the output file by the column that represents your output variable. Find the minimum, maximum, and mean.

(c) Do your data match the expectations that you articulated in point (a)? How do they compare with the data you collected for the next-generation model? □

Wow! What is going on here? Let us think carefully. We chose settings where $\alpha = \beta$. This means that the removal rate α for host j^* is exactly the same as the rate β at which j^* makes contact with a given adjacent host i. Thus, there should be a fifty-fifty chance that the first effective contact between hosts j^* and i occurs *before* the time $T_{j^*}^R$ of removal of the index case. In other words, we should have $p = 0.5$ and a mean number of about 51 hosts who will experience infection. In general, for continuous-time models it can be shown that

$$p = \frac{\beta}{\alpha + \beta}. \tag{9.16}$$

Exercise 9.78. Derive Eq. (9.16). □

From Eq. (9.16) we can derive the following approximate formula for R_0 in continuous-time models:

$$R_0 \approx \frac{\beta}{\alpha + \beta}\langle k \rangle. \tag{9.17}$$

This is in fact the formula that IONTW uses for continuous-time models when you click **Metrics**. If the network is a tree, then the right-hand side of Eq. (9.17) gives us the exact value. In other types of networks, the formula may only be approximately true, as we will see shortly.

The alert reader will have noticed a discrepancy between Eq. (9.17) and the formula (9.4) for R_0 that we gave in Section 9.2.2 of the main text. Both Eqs. (9.17) and (9.4) are approximations, but which one is better?

Equation (9.17) is the more accurate approximation. When studying compartment-level models we usually assume that the population size N is very large. The contact network corresponds to the complete graph K_N. Under these assumptions, the difference between N and $N - 1 = \langle k \rangle$ becomes negligible, and we usually have $\beta \ll \alpha$, so that $\frac{\beta}{\alpha+\beta} \approx \frac{\beta}{\alpha}$. Then the right-hand side of Eq. (9.17) becomes approximately equal to $\frac{\beta N}{\alpha}$, which is the approximation given by Eq. (9.4).

For discrete-time models with parameters a and b, we obtain the following version of Eqs. (9.16) and (9.17):

$$\begin{aligned} p &= \frac{b}{a + b - ab}, \\ R_0 &\approx \frac{b}{a + b - ab}\langle k \rangle. \end{aligned} \tag{9.18}$$

Exercise 9.79. Derive Eq. (9.18). □

OK, but where does the huge variability that we observed in Exercise 9.77 come from? Let us first see whether this is something that occurs only in continuous-time models. Change the following parameter settings:

model-time → Discrete
infection-prob: 0.3333
end-infection-prob: 0.5

Because $a < 1$, this discrete-time model is no longer a next-generation model. According to Eq. (9.18), we should get the same values of p and R_0 as for the continuous-time model that we just explored.

Exercise 9.80.

(a) Set up and run a batch processing experiment as for Exercise 9.76. What would you expect in terms of the output column?
(b) Sort the output file by the column that represents your output variable. Find the minimum, maximum, mean, and standard deviation.
(c) Do your data match the expectations that you articulated in point (a)? How do your data compare with those that you found for Exercises 9.76 and 9.77? □

Exercise 9.81. Try to find a plausible explanation for the differences in variability that you observed between Exercises 9.76, 9.77 and 9.80. Be sure to look up our sample solution and compare notes. □

Now let's revisit our old friends, the complete graphs that we used in the simulations for Chapter 8 and Section 9.2. Change the following parameter settings if need be:

model-time → Continuous
infection-rate: 0.6
end-infection-rate: 0.6
network-type → Complete Graph
num-nodes: 10
set-state-by → Number of nodes
num/frac: 1
auto-set: On

Create a network and click **Metrics**. Look up the value of `R0` in **Command Center** and check whether it matches the right-hand side of Eq. (9.17). Set the speed slider near the end of the slow range; adjust for comfortable viewing as needed.

We will do the following: Click **Go** and watch the movie in slow motion, and click **Go** again the moment the index case turns gray. Then record the total number of nodes that experienced infection. Try it out first to adjust the speed for comfort, then repeat 10 times and record the numbers of nodes that are either red or gray at the moment the index case got removed. Next calculate the mean and subtract 1 from it to account for the index case. One might expect that the mean you computed would give us a reasonably good estimate of R_0.

Exercise 9.82. Did you get a higher or a lower value than the one you found using **Metrics**? If you noticed a discrepancy, how would you explain it? □

OK, the mean number of hosts that are either infectious or removed by time $T^R_{j^*}$ may not give a good estimate of $R_0 + 1$. Is this a big deal? Perhaps we just used a bad method for empirically estimating this parameter?

Actually, the method is a bad one, but along the way you may have discovered something important. Before reading on, be sure to check our sample solution for Exercise 9.82 and compare notes.

Exercise 9.83.

(a) Explain why, in general, the right-hand sides of (9.17) and the second line of (9.18) are only approximations. Would they give overestimates or underestimates?
(b) Show that the estimate (9.18) gives the correct value of R_0 in next-generation models.
(c) Show that the estimates give the correct value for R_0 if the contact network is a tree or forest, regardless of how we model time.
(d) Show that these estimates become arbitrarily good when the contact network is K_N and $N \to \infty$. □

9.6 MODULE 9.6: EXPLORING ERDŐS-RÉNYI RANDOM GRAPHS WITH IONTW

9.6.1 The Structure of Erdős-Rényi Random Graphs

Open IONTW, click **Defaults**, set the speed control slider to the extreme right, and use the following parameter settings.

network-type → Erdos-Renyi
num-nodes: 200
lambda: 3
auto-set: Off

Click **New** to create an instance of $G_{\mathrm{ER}}(200, 3)$. Look at the degree distribution in the **Network Metrics** plot. Then click **Metrics** and look up the value of the mean degree in the **Command Center**. Record your observations and repeat a few times. You may want to vary **num-nodes** and **lambda**; be sure to also look at some graphs $G_{\mathrm{ER}}(N, \lambda)$ with $\lambda = \frac{N}{2}$.

Exercise 9.84. Do your observations confirm the predictions about the degree distribution that we made in Section 9.3.5? □

Now set
num-nodes: 300
lambda: 1.5

Next, click **Set** to introduce one infectious node, and find the connected component of this node by using the trick that you discovered in Exercise 9.60. Repeat about 10 times for this network using **Reset** and then **Set**. This will keep the network fixed, but will change the initially infectious node. Record the approximate sizes of the connected component by moving your mouse over the relevant part of the red curve in the **Disease Prevalence** plot.

Exercise 9.85. What do you observe? Do you get connected components with a range of different sizes? If the component is large, is it always the same one? How can you tell from the plot? □

The results may look puzzling. Repeat 10 more times, but look at a different instance $G_{\mathrm{ER}}(300, 1.5)$ each time by using **New** instead of **Last** for initializing the network.

Exercise 9.86. In what respect are the results similar to the ones of the previous exercise? In what respect are they different? □

This is interesting. It appears that there is always one very large component in addition to many small ones. Mathematicians can prove that this will indeed be the case for instances of $G_{\mathrm{ER}}(N, \lambda)$ with probability very, very close to 1, as long as $\lambda > 1$ and N is large. The large component is called the *giant* component. More detailed explorations of the connected components of $G_{\mathrm{ER}}(N, \lambda)$ with IONTW can be found in the modules of [6]. The survey paper [7] gives an overview of the theoretical results.

9.6.2 Disease Transmission on Erdős-Rényi Random Networks

Here we will run some simulations of disease transmission on Erdős-Rényi contact networks and compare them with the predictions of compartment-level models. In all simulations we will choose parameters that give the same value of R_0.

Open IONTW, click **Defaults**, and use the following parameter settings:
model-time → Continuous
infection-rate: 1
end-infection-rate: 1
network-type → Erdos-Renyi
num-nodes: 100
lambda: 4
auto-set: On

Check that the value for R_0 that you get from Eq. (9.17) is the same one you get by clicking **New** and then **Metrics**. Set up and run a batch processing experiment with the following settings:
Define a **New** experiment.
Repetitions: 100
Measure runs using these reporters:

```
count turtles with [removed?]
```

Setup commands:

```
new-network
```

Save your output for subsequent analysis and change the following parameters:
infection-rate: 0.0202
network-type → Complete Graph

Check that the value for R_0 that you get from Eq. (9.17) is the same one you get by clicking **New** and then **Metrics**, and that it is very close to the one in the previous experiment.

Thus we will be comparing two *continuous-time* disease transmission models with the same R_0, one on an Erdős-Rényi random network, and one with the uniform mixing assumption. According to what we wrote in the main text, we might expect that the results should be similar. Set up a **New** batch processing experiment for the model that we just defined, and run it.

Now change

model-time → **Discrete**

infection-prob: 0.02

end-infection-prob: 1

Check that the value for R_0 that you get from Eq. (9.17) is the same one you get by clicking **New** and then **Metrics**, and that it is very close to the one in the previous experiment. Set up a **New** batch processing experiment for the model that we just defined, and run it.

Finally, change

network-type → **Erdos-Renyi**

infection-prob: 0.5

Check that the value for R_0 that you get from Eq. (9.17) is the same one you get by clicking **New** and then **Metrics**, and that it is very close to the one in the previous experiment.

Thus we will be comparing two *next-generation* disease transmission models with the same R_0, one on an Erdős-Rényi random network, and one with the uniform mixing assumption. According to what we wrote in the main text, we might expect that the results should be similar. Set up a **New** batch processing experiment for the model that we just defined, and run it.

Exercise 9.87.

(a) Analyze the data that you collected in the above experiments. Sort the output columns from smallest to largest values. Classify observed outbreaks into minor and major ones. Record the numbers of minor and major outbreaks and also the number of runs where no secondary infections whatsoever occurred. Record the maximum size of minor outbreaks and both the maximum and minimum sizes of major ones. Compute mean numbers of hosts who experienced infection for minor and major outbreaks separately.

(b) Are the results for the 4 models similar enough that we could attribute all differences to random fluctuations?

□

How can the similarities and differences between the outcomes of the simulations that you ran for Exercise 9.87 be explained? This intriguing question goes beyond the scope of this chapter. You can find the answer in the modules posted at [6].

REFERENCES

[1] White LF, Wallinga J, Biggerstaff M, Cauchemez S, Reed C, Gambhir M, et al. Estimation of the reproductive number and the serial interval in early phase of the 2009 influenza A/H1N1 pandemic in the USA. Influenza Other Resp Viruses 2009;3:267-276.

[2] Centers for Disease Control and Prevention. Measles http://www.cdc.gov/measles/. Accessed: 2014-08-14.

[3] Anderson RM, May RM. Directly transmitted infectious diseases: control by vaccination. Science 1982;215:1053-1060.

[4] Centers for Disease Control and Prevention. Diphtheria http://www.cdc.gov/diphtheria/. Accessed: 2014-08-14.

[5] Just W, Callender H, LaMar MD, Toporikova N. Online Appendix: Transmission of infectious diseases: data, models, and simulations. 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.

[6] Just W, Callender H, LaMar MD. Exploring transmission of infectious diseases on networks with NetLogo. Accessible at https://qubeshub.org/iontw and also http://www.ohio.edu/people/just/IONTW/.

[7] Spencer J. The giant component: the golden anniversary. Not. Am. Math. Soc. 2010;57:720-724.