<1>Chapter 7: Modeling non-communicable diseases

In previous Chapters, we used Markov models to estimate the burden of disease, and the potential impact of our interventions. One of the key limitations to Markov models is that they don't take into account a person's unique individual characteristics; Markov models are designed to efficiently simulate the average outcome for an entire population. For many public health and healthcare system problems, however, we need to consider *heterogeneity* within a population, or differences in risk and differences in benefit from our programs. For that purpose, *microsimulation* models, which take into account unique characteristics of individuals and the correlations between these characteristics, can be more useful. In this Chapter, we detail the constructing and use of microsimulation models, using examples related to diabetes prevention and treatment.

<2> How to screen for type 2 diabetes?

In 2013, Dr. Akihiro Seita of the United Nations' Relief and Works Agency (UNRWA) sent out an email to diabetes experts around the world. Dr. Seita was in charge of a massive UN system of hospitals and clinics that care for refugees in the Middle East. In total, he oversaw 3,000 staff at 138 hospitals and clinics serving 5 million refugees.

Although Dr. Seita had dealt with numerous complex problems—clean water delivery, transportation problems in conflict zones, and infectious disease outbreaks—he had recently encountered a problem he hadn't anticipated: the rising prevalence of type 2 diabetes among refugees.

In his email to diabetes experts, Dr. Seita explained that numerous refugees were coming to hospitals and clinics with complications of diabetes, such as blindness, stroke, and kidney failure. It was thought that the prevalence of the disease might be as high as 1 in 5 adults; but few people were diagnosed in time to avoid the complications of the illness. The refugees' diet might have been part of the issue, as humanitarian food packages were often loaded with carbohydrates.

In response to his email, Dr. Seita received dozens of suggestions about what to do. Most of the experts suggested screening people in the community for diabetes. Some experts thought Dr. Seita should widely administer a blood test for diabetes, which is a fasting (early morning, before eating) blood sugar test. Those people with fasting blood sugar $\geq$126 mg/dL are said to have diabetes. But Dr. Seita's budget was so limited that he couldn't administer many blood tests. He needed a survey that could be cheaply administered to a large population of people, to find those at highest risk for diabetes; those at high risk, according to the survey, could be sent to health clinics to get a blood test.

Finding a survey to screen for diabetes risk was not a problem; there were dozens of surveys available. The question was: which survey might best serve Dr. Seita's population? Different surveys used different criteria to determine who was considered "high risk" for diabetes. One survey suggested that everyone with a waist circumference greater than 80 centimeters should be considered "high risk". Another survey suggested that a combination of two factors would be necessary to be considered "high risk": a waist circumference greater than 80 centimeters, and a blood pressure of at least 140 mmHg (systolic pressure). Yet a third survey suggested that only those people with a blood pressure of at least 150 mmHg and with a body mass index (kilograms of weight divided by meters-squared of height) of at least 30 kg/m^2 should be considered "high risk", regardless of their waist circumference.

Which screening instrument should Dr. Seita choose? Dr. Seita ultimately wanted to use whichever screening tool let him find the most people with diabetes (a highly-sensitive test) and get the fewest "false-positive" screening test results (a highly-specific test). The goal was to detect people with diabetes in time to give them treatment and avoid complications of the disease.

<2> The motivation for microsimulation

Suppose that we wanted to answer Dr. Seita's question using a Markov model. A typical Markov model of type 2 diabetes might have just two states: a healthy state for people without diabetes, and a state for people with diabetes (Figure 7.1). The flow between the two states would be the rate of diabetes in the population. But how would we be able to use such a model to compare the screening tools? The model would only have the average rate of diabetes in the population. Those people with diabetes would be the group that the screening surveys should find; the more of these people who screen positive, the more sensitive the test. The fewer people who are healthy who screen positive, the more specific the test. But how do we know which subset of people within each state would test positive? The Markov model, as it is drawn in Figure 7.1, can't tell us anything about individual features of the people in each state; it just lumps all people together based on whether they are healthy or have diabetes.

[INSERT FIGURE 7.1 HERE].

If we wanted to use the Markov model to capture how different screening tools might work in the refugee population, perhaps one strategy would be to break up each of the states into different populations of people: those with a large waist circumference, those with high blood pressure, and those with high body mass index, for example. Such a model is shown in Figure 7.2. As shown here, we can now try to estimate what the rate of diabetes is among people with a large waist circumference, what the rate of diabetes is among people with high blood pressure, and what the rate of diabetes is among people with high body mass index. Figure 7.2 shows some typical rates from Dr. Seita's population. One screening survey would people with high waist circumference. Another would affect people with high waist circumference and high blood pressure. A third instrument would affect the people with a different definition of high blood pressure, and high body mass index.

[INSERT FIGURE 7.2 HERE].

Despite making our Markov model more complex, we still face major problems with utilizing it to identify the best screening instrument: there will be people who fall simultaneously into many of these states. For example, some people will have all three risk factors (high waist circumference, high blood pressure, and high body mass index). Presumably, these people will be at the highest risk for diabetes. Other people will have two of the three, and some people will have just one risk factor or no risk factors. Also, it's unclear how we should define "high blood pressure" when it's a continuous measure and one instrument uses a cutoff of 140 mmHg to define "high blood pressure" while another uses a cutoff of 150 mmHg to define "high blood pressure".

A natural next step is to try to make a Markov model that has states for every possible combination of features: a state for people with just high waist circumference, or both high waist circumference and high blood pressure, or with various combinations of states that can account for both definitions of high blood pressure. Very quickly, we would have a large number of different states, and an unmanageably complex model.

The natural alternative to making such a complex model would be to ultimately make enough states that we can capture every combination of characteristics we would want to simulate. We can reduce the model all the way from the broad population to each individual, taking into account each individual's unique risk factors within the population. A model that takes into account the unique properties of each individual is known as a *microsimulation.* Rather than just describing a group of people and what state they are in (a Markov modeling approach), a microsimulation is like a large table describing each person in the population. As demonstrated in Table 7.1, an individual in the table may have one or more risk factors, and all 5 million people might be considered in the model.

[INSERT TABLE 7.1 HERE]

<2> Constructing a microsimulation

To construct a microsimulation, there are two tools that we need: (i) an estimate of the distributions of characteristics we want to simulate (such as their means and standard deviations); and (ii) an estimate of how correlated those characteristics are. Dr. Seita provided us with these estimates: in a brief survey of a subsample of his refugee population, people had a mean waist circumference of 75cm (standard deviation, sd = 10 cm), a mean blood pressure of

130mmHg (sd = 25 mmHg), a mean body mass index of 25 kg/m^2 (sd = 5 kg/m^2), and a fasting blood sugar of 108 mg/dL (sd = 20 mg/dL). Table 7.2 shows the correlation among these characteristics as a *correlation matrix*, which is a set of Pearson correlation coefficients capturing the relationship between each factor and every other factor (the coefficient is capturing how often each set of factors is found commonly with each other set of factors, i.e., higher waist circumferences are positively correlated with having higher blood pressure, which are positively correlated with having higher body mass index, which are positively correlated with higher fasting blood sugars).

[INSERT TABLE 7.2 HERE]

It's important to take into account the correlation between factors because our screening survey instruments are trying to capture the highest-risk people, who presumably have two or three of the risk factors of high waist circumference, high blood pressure, or high body mass index.

Next, we need to turn on *RStudio* (see Chapter 7) and use our two pieces of information —the distribution (mean and standard deviation) of each characteristic, and the correlation among characteristics—to construct a simulated refugee population.

To construct our population, we open a new code window and insert the input parameters we need to provide this population with the key characteristics for each individual. This example will illustrate the power of *R*: there is no way, to our knowledge, to construct such a microsimulation in Excel. The full code for the subsequent microsimulation is also available for download on our textbook website.

In our code, first let's declare the sample size we want to simulate:

```
n = 5000000
```

Here, variable *n* is being given a value of the 5 million, the number of people we want to simulate.

Next, we need to input the prevalence and standard deviation (uncertainty around the prevalence) of each of the four characteristics we're trying to include in our simulation: high waist circumference, family history, high blood pressure, and fasting blood sugar. Let's create a vector of prevalence of characteristics and another vector of standard deviations around those prevalence estimates:

```
chars=c(75, 130, 25, 108)
sds=c(10, 25, 5, 20)
```

Here, we have the characteristics in the order stated: waist circumference, blood pressure, body mass index, and blood sugar.

Our next step is to insert our correlation matrix:

```
cormat = matrix(c( 1.000,0.302, 0.571,  0.776,
                   0.302, 1.000, 0.110,  0.587,
                   0.571, 0.110, 1.000 , 0.002,
                   0.776,0.587, 0.002, 1.000),ncol=4,byrow=TRUE)
```

Just as described in Chapter 7, we have a matrix with four columns, read row by row from left to right.

The critical next step is to use these two pieces of information—distribution of characteristics and correlation between them—to generate a simulated population. To do this, we need to make use of a key feature of *R*: the ability to install packages that external users have created, to save us time and do tasks that are not built-in to *R* by default. Installing packages is like installing apps on a smartphone or adding a turbo drive to a car engine—they help us do new things, and can rapidly expand our capabilities. We'll install two packages for microsimulation by typing in the following commands:

```
install.packages('MASS')
install.packages('MBESS')
library(MASS)
library(MBESS)
```

The first two lines (install.packages) only have to be run once; the packages are then installed on your hard drive and don't need to be run again. The next two lines (library) tell *R* to activate the packages, just like clicking and starting an app on a smartphone. They need to be run every time we start up the program.

Now that the two packages are installed and activated, we can use them. We first want to

create a Table similar to Table 7.1, in which the prevalence rates and correlations between characteristics in the refugee population can be translated into a simulation of people and their unique characteristics. We will need to convert the correlation matrix into a *covariance matrix*, which is simply a translation of correlation statistics into a scaled and standardized format; the covariance reflects how much the factors vary together, taking into account their standard deviations, and can be calculated from the correlation matrix using the MBESS package in *R*:

```
sigma=cor2cov(cormat,sds)
```

Here, we use the cor2cov function in the MBESS package to convert the correlation matrix to a covariance matrix.

Next, we take into account both the prevalence and the covariance matrix to simulate a population of 5 million people. We create a sample of characteristics that simultaneously matches the distribution of each individual characteristic in the population (its mean and standard deviation) and the correlation between characteristics in the population (the correlation matrix). To do so, we use the MASS package in *R*:

```
pop=mvrnorm(n,chars,sigma)
```

We specifically create a new matrix *pop* using the mvrnorm function in the MASS package, which takes the number of samples we want (n = 5 million), the means (*chars* vector) and the covariance matrix (*sigma* matrix), and—after a moment of computation—produces a table of 5 million people and their characteristics that is similar to Table 7.1. The function creates a multivariate (many variables) normal distribution of each characteristic to match the distribution and covariance matrix for the population. If non-normal or other atypical distributions are called for, the Help function in *RStudio* provides a guide to sampling from many other types of distributions.

Once we have created our matrix of people, we can take a look at what it looks like. We might not want to view the whole table at once, since it's 5 million rows long for the 5 million people we have simulated. We can see the first 10 rows by using the *R* command head:

```
> head(pop, 10)
          [,1]     [,2]     [,3]      [,4]
[1,] 71.94014 114.0875 26.52497  92.55315
[2,] 64.61419 121.3492 18.49197 100.20444
[3,] 74.93484 163.1153 26.97546 110.30267
[4,] 81.88704 145.4949 30.98941 112.00721
[5,] 87.48825 146.8071 21.51043 146.29444
[6,] 82.96516 144.4435 33.76540 106.99635
[7,] 70.59627 131.9912 25.55418  97.47015
[8,] 71.65715 118.3817 20.14378 109.57800
[9,] 90.48061 143.9922 30.41948 128.85458
[10,] 85.66734 165.6580 34.77085 115.84608
```

We see that in column 1, we have a series of waist circumference values; in column 2, we have a series of blood pressure values; in column 3 we have a series of body mass indices, and in column 4 we have fasting blood sugar values.

Of course, because we're doing a simulation with random numbers, the exact numbers will change from simulation to simulation. But what we hope to achieve is to fairly compare our screening instruments in a simulated large population, such that on average we will get the right means and distributions and correlations as the real population (per the Central Limit Theorem of statistics). By simulating a large population, we can hope to identify how certain or uncertain we are about the comparison between the three screening instruments. We can even perform the whole simulation multiple times (as we did in Chapter 7), to capture variations in our outcome based on uncertainty in our input parameter values.

How do we know that our simulation has correctly captured the distribution of characteristics in our population and the correlation between characteristics? We can use a few commands to ensure that our simulation has matched our two key pieces of information:

```
install.packages('matrixStats')
library(matrixStats)
colMeans(pop)
colSds(pop)
cor(pop)
```

Here, we installed a package called *matrixStats*, which allows us to quickly calculate statistics on matrices. We then activated the package using the library command, and then

statistics on matrices. We then activated the package using the library command, and then checked that the means, standard deviations, and correlation matrix of our resulting matrix *pop* corresponded well to our input data. The colMeans command takes the means of each column of the matrix, the colSds command takes the standard deviations of each column, and the cor command gives us the correlation matrix.

We can even plot the values of characteristics and visualize their correlation:

```
pairs(~pop[1:1000,1]+pop[1:1000,2]+pop[1:1000,3]+pop[1:1000,4])
```

Whenever calling elements of a matrix in *R*, square brackets tell *R* which row and which column to refer to; so [1,2] means row 1, column 2, while [2,] means row 2, all columns, and [,3] means all rows for column 3. Hence, in the above command, we've asked *R* to plot all pairs of values for the first 1,000 rows of each column against each other; we only chose to plot 1,000 values because 5 million values take a lot of processing power and can't be easily visualized. The resulting plot is in Figure 7.3.

[INSERT FIGURE 7.3 HERE]

<2> Using the microsimulation to compare interventions

Now that we have constructed our simulated population, we can use it to compare our three diabetes screening survey instruments.

One survey suggested that everyone with a waist circumference greater than 80 centimeters should be considered "high risk". A second survey suggested that a combination of two factors would be necessary to be considered "high risk": a waist circumference greater than 80 centimeters, and a blood pressure of at least 140 mmHg (systolic pressure). Yet a third survey suggested that only those people with a blood pressure of at least 150 mmHg and with a body mass index (kilograms of weight divided by meters-squared of height) of at least 30 kg/m^2 should be considered "high risk", regardless of their waist circumference.

Suppose that we wanted to find the instrument with the highest sensitivity and highest specificity for finding people with diabetes.

First, we need to convert the fasting blood sugar values to a dichotomous variable for whether or not a person has diabetes:

```
dm = (pop[,4]>=126)
```

Here, we ask *R* whether the value in column 4 of the *pop* matrix is greater than or equal to 126 mg/dL; if so, *R* returns a value of 1, and otherwise returns a value of 0. Using the above command, we've created a vector that is 5 million people long, with 0's and 1's to tell us whether or not a person has diabetes. We can get some statistics on this variable by typing mean(dm) to see that on average about 18% of people have diabetes.

Now let's see how many people will test positive for survey 1 (waist circumference >80cm), survey 2 (waist circumference >80cm and blood pressure≥140 mmHg) and survey 3 (blood pressure ≥150 mmHg and body mass index ≥30 kg/m^2). We can create three variables corresponding to the number of people positive or negative on each of the three surveys:

```
survey1pos = (pop[,1]>80)
survey2pos = (pop[,1]>80)&(pop[,2]>=140)
survey3pos = (pop[,2]>=150)&(pop[,3]>=30)
```

We have asked *R* to put in 1's and 0's into each of the three variables depending on whether or not each row (each individual's data) corresponds to testing positive for each screening test. Notice that *R* allows us to rapidly process the entire 5-million-person-long row of data in just one command.

We can quickly tabulate how many people tested positive with each survey tool by using the table command:

```
> table(survey1pos)
survey1pos
 FALSE    TRUE
3457967 1542033
> table(survey2pos)
survey2pos
 FALSE    TRUE
4265047  734953
> table(survey3pos)
survey3pos
```

```
 FALSE    TRUE
4790793  209207
```

If we want to see what percentage tested positive, we just have to divide by the total population size, *n*:

```
> table(survey1pos)/n
survey1pos
   FALSE      TRUE
0.6915934 0.3084066
> table(survey2pos)/n
survey2pos
   FALSE      TRUE
0.8530094 0.1469906
> table(survey3pos)/n
survey3pos
   FALSE      TRUE
0.9581586 0.0418414
```

Finally, we can calculate the sensitivity and specificity of each survey instrument. The sensitivity is the number of people testing positive divided by all people with the condition, while the specificity is the number testing negative divided by all people without the condition.

The sensitivity can be translated into *R* code as the following:

```
sens1 = sum((survey1pos==1)&(dm==1))/sum(dm==1)
sens2 = sum((survey2pos==1)&(dm==1))/sum(dm==1)
sens3 = sum((survey3pos==1)&(dm==1))/sum(dm==1)
```

Here, we determine the total (sum) number of people that are both positive in testing by a survey instrument, and have diabetes, divided by the total testing positive.

Analogously, the specificity can be coded as:

```
spec1 = sum((survey1pos==0)&(dm==0))/sum(dm==0)
spec2 = sum((survey2pos==0)&(dm==0))/sum(dm==0)
spec3 = sum((survey3pos==0)&(dm==0))/sum(dm==0)
```

Comparing the outcomes, we see that the sensitivities and specificities are:

```
> sens1
[1] 0.8050339
> sens2
[1] 0.5482468
> sens3
[1] 0.09512384
> spec1
[1] 0.8037088
> spec2
[1] 0.9433038
> spec3
[1] 0.9704223
```

We see that overall, survey instrument 1 is best from the perspective of maximizing sensitivity (at 81% sensitivity), and it has 80% specificity as well. Survey instrument 2 is unimpressive in terms of sensitivity (55%), but has higher specificity (94%). Survey instrument 3 has terrible sensitivity (only 10% sensitivity), but is best from the perspective of preventing false positives (specificity 97%).

Ultimately, none of the three instruments are very impressive, but we could use our model to test alternative instruments of combinations of instruments to improve performance. One common way that epidemiologists compare the performance statistics of tests is to plot a "receiver operating characteristic" (ROC) curve. The ROC curve plots 1 minus the specificity of an instrument against the sensitivity of an instrument, and a *C-statistic* is calculated as the area under the curve. Higher *C*-statistics (closer to 1) indicate a better ability of an instrument to discriminate people with the disease from those without.

We can use the Epi package in *R* to calculate the *C* statistic:

```
install.packages('Epi')
library(Epi)
ROC(survey1pos, dm)
```

The ROC function requires us to first type the test results, then the true disease status, in parentheses. The ROC curve for instrument 1 is shown in Figure 7.4, which reveals a *C*-statistic of 0.804. Typically, we aim for *C*-statistics that are over 0.8 for an instrument to be considered

of 0.804. Typically, we aim for C statistics that are over 0.8 for an instrument to be considered strong, but for screening purposes we would favor a high-sensitivity instrument over a high-specificity instrument to detect as many people as possible who have the disease.

[INSERT FIGURE 7.4 HERE]

<2>Addressing decision uncertainties using microsimulation

Our previous example of microsimulation permitted us to examine how microsimulation can help us detect individuals at different degrees of risk for an outcome, overcoming a hurdle of Markov modeling that tends to focus on the population average rate of disease.

Another important advantage of microsimulation is its ability to allow us to make better decisions when we face a lot of uncertainty. Let's take the following example: suppose we have a medical clinic for which we need to make a hiring decision. Can we afford to hire a new nurse to help us at the clinic?

Currently, we have five doctors at the clinic who see on average 110 patients per day among them (standard deviation of 15), for every day of the 240 days per year that the clinic is open. Because different patients have different levels of illness, they take different amounts of time. Furthermore, because different patients have different insurance companies and require different levels of service, the clinic gets paid different amounts per patient, with the typical patient visit producing $100 in gross revenue (standard deviation of $20). Suppose the clinic currently employs a number of nurses, medical assistants, and receptionists, as well as paying for rent, equipment, and utilities. In total, each year, the collective salary and overhead costs at the clinic total about $2,600,000.

The key question is whether the clinic can afford to hire a new nurse to help around the clinic. The nurse can help make the clinic more efficient, so that on average another 5 patients per day might be seen. But the nurse also costs $100,000 in salary.

To solve this problem, we can use our microsimulation methods to identify whether the clinic may be able to afford the expense of hiring a new nurse, while taking into account the numerous uncertainties along the way: the uncertainty in how many patients the doctors at the clinic will see on different days, the uncertainty in how much the clinic gets paid for those visits, and the uncertainty in how much more efficient the nurse will make the clinic.

Let's start by putting in the data we have available for the base case condition (before hiring an additional nurse), starting with the number of days the clinic is open:

```
daysperyr = 240
```

We can then estimate the typical visits per year at the clinic, which can be simulated by sampling the number of visits per day (mean = 110, sd = 15) for each of the 210 days per year:

```
visitsperyr = rnorm(n=daysperyr,mean=110,sd=15)
```

Next, we can calculate the total gross revenues at the clinic by sampling from the revenue per visit (mean = $100, sd=$20), for each of the visits per year at the clinic:

```
revpervisit = rnorm(n=visitsperyr,mean=100, sd=20)
```

Finally, we can add up the total expected gross revenue for the clinic per years:

```
grossrev = sum(visitsperyr*revpervisit)
```

We can determine the current net, after subtracting $2,600,000 in salary and overhead costs from our gross revenue:

```
netrev = (annualrev-2600000)
```

Next, we can repeat the process after taking into account the increased revenue from adding our nurse, which is from adding another 5 patients per day, and subtracting the additional $100,000 in salary:

```
visitsperyr = rnorm(n=daysperyr,mean=110+5,sd=15)
revpervisit = rnorm(n=visitsperyr,mean=100, sd=20)
grossrev = sum(visitsperyr*revpervisit)
netrev = (annualrev-2600000)-100000
```

The variable *netrev* tells us how much money is left over if we hired the additional nurse; it will be positive if we can afford the nurse, and negative otherwise. As shown here, we just sample one time for one year's worth of simulated visits and revenues. We want to determine how often we might be able to safely hire a nurse, so we should repeatedly sample from the distributions of uncertainty to determine how often the *netrev* result is a positive number.

We can choose an arbitrarily-large number of times (let's say 100,000) to repeat the entire simulation and plot how many times we got a positive revenue outcome. One way to do that is to create an empty vector of zeros to store our results from each of the 1 million simulations, and

create an empty vector of zeros to store our results from each of the 1 million simulations, and then fill in the results using a "for loop" as in Chapter 7:

```
revresults=rep(0,100000)
for (i in 1:100000){
 daysperyr = 240
 visitsperyr = rnorm(n=daysperyr,mean=110+5,sd=15)
 revpervisit = rnorm(n=visitsperyr,mean=100, sd=20)
 grossrev = sum(visitsperyr*revpervisit)
 netrev = (grossrev-2600000)-100000
revresults[i] = netrev
}
```

Finally, we can plot the results as a histogram, as shown in Figure 7.5.

[INSERT FIGURE 7.5 HERE]

We can see that we're not 100% certain that we'll always have a positive revenue outcome. What proportion of the time did we end up with a positive result? We can tabulate the percent of the time that our result was positive

```
> table(revresults>0)/100000

 FALSE    TRUE
0.08058 0.91942
```

It looks like about 92% of the time, of our 100,000 simulations, the clinic was able to afford the nurse.

This example shows how we can use *R*'s ability to perform repeated large-scale simulations, and repeated sampling from probability distributions, to track uncertainty in a decision and get a sense of whether the decision might be wise or not. In most circumstances, we don't have definite confidence that our mean results are reflective of what will happen every day, month or year. In uncertain contexts, *R* can help us plan our budgets and resource allocations with a better appreciation for the risks and benefits we might experience.

<2> Agent-based modeling: an extension of microsimulation

An increasingly-popular extension of microsimulation is known as *agent-based modeling*. Agent-based models simulate individuals, but have a critical feature that standard microsimulation models do not: they include equations to describe the *behavior* of individuals, not just their fixed features like waist circumference or body mass index. For example, an agent-based model of type 2 diabetes might incorporate how much a person drinks sugary beverages, and include equations to describe how much higher the risk of drinking such beverages might be if a person is surrounded by other people who also drink sugary beverages (e.g., social contagion).

To extend our diabetes example, we can move from the Middle East to the other side of the world—to the country of Mexico—where behavioral responses to diabetes risk factors have become a major subject of national concern. In recent years, the epidemiologist Juan Dommarco at the National Institute for Public Health in Mexico studied strategies to reduce the risk of type 2 diabetes among young people drinking sugary soda drinks. Dr. Dommarco was asked to estimate what impact the central government might have on diabetes risk if drinking soda did not continue to increase in popularity, but was substituted with clean drinking water and was made less popular through marketing campaigns and taxes on soda imposed by the government.

The textbook website links to *R* code that describes an agent-based model of sugary soda consumption. The model of drink consumption has features common to many agent-based models. It can be constructed much like a microsimulation, but with an extra step: in addition to simulating the population, and assigning fixed attributes to that population, we additionally set behavioral "decision rules" that describe how people act and interact with each other. Individuals decide, in our case, whether to drink soda based in part on how many others around them are drinking soda.

We use a classical equation for simulating product consumption in a population over time (called the Bass diffusion model; note that we present a version of the model commonly used in public health, although numerous other versions also exist with additional complexities) [1]. The model can be described by Equation 8.1:

[Equation 8.1]

In Equation 8.1, *P(t)* is average probability that a person consumes soda in year *t*, *m* is the typical maximum fraction of the population that will be willing to consume soda at some point, and *q* is a "social diffusion" parameter describing how easily a person is influenced to consume soda by their peers. Based on fitting the model to data from Mexico, we found typical values for the parameters *q* and *m* for the population in Mexico, which are inserted into the *R* code (*q* = 0.1159, *m* = 1). The intuition behind the model formula is that the change in the probability of consuming soda is proportional to some degree of social influence *q*, as well as a logistic curve (Figure 7.6), such that consumption becomes popular initially and then slows in popularity as the probability of consumption reaches the maximum consumption level *m* in the population.

[INSERT FIGURE 7.6 HERE]

To create the model, we first begin by specifying our parameters:

```
n=100000
q=0.1159
m=1
rr=1.4
dmrisk=0.01
p=0.4
time=20
```

Here, we are simulating a representative 100,000 people (*n*) with a given set of social diffusion parameters. We also specify that drinking soda confers a relative risk (*rr*) of diabetes of 1.4 as compared to people not drinking soda, and that the baseline (non-soda-drinker) risk of diabetes each year is 0.01 (1%). We specify that the starting probability of drinking soda is 40% (*p*) and we want to simulate a 20-year time period.

Next, we create vectors that use the binomial probability function (rbinom) to determine whether people at the beginning of the model are drinking soda or not, and whether people have diabetes, or not. The binomial probability function acts like a weighted coin, where the chances of flipping to "heads" (which means the person will consume soda, or has diabetes) is given by a probability. The function rbinom creates a vector of length *n*, flipping a coin once per each person, with probability p of drinking soda (or 0.15 of having diabetes):

```
soda = rbinom(n,1,p)
dm = rbinom(n,1,0.15)
```

We have to create some vectors to keep track of the total number of soda drinkers and diabetes cases over time:

```
totsoda = sum(soda)
totdm = sum(dm)
```

We then simulate the rate of diabetes conditional on whether someone drinks soda, taking into account the higher rate of diabetes among people who are soda drinkers. Specifically, in each time step of our "for loop" from time point 2 to 20, we will first calculate the probability of soda consumption. We then create the vector *soda* by sampling from a binomial probability function to determine whether people that year are drinking soda or now. We create vector *newdm* to then simulate their risk of diabetes, taking into account the relative risk of diabetes if drinking soda. Finally, we update our vector *dm* to account for new diabetes cases. Note that because diabetes is irreversible, we only update the subset of cases that don't yet have diabetes `(dm==0)*newdm`. We keep track of the total soda consumers and total cases of diabetes in our vectors *totsoda* and *totdm*:

```
for (i in 2:time){
 p[i] = p[i-1] + (q/m)*p[i-1]*(m-p[i-1])
 soda = rbinom(n,1,p[i])
 newdm = rbinom(n,1,dmrisk*rr*soda)
 dm = (dm==1)+(dm==0)*newdm
 totsoda = c(totsoda,sum(soda))
 totdm = c(totdm, sum(dm))
}
```

We see that the viral social phenomenon of drinking soda could dramatically increase the prevalence of diabetes over the simulated period:

```
> totdm
[1] 15055 15551 16086 16646 17243 17867 18532 19226 19922 20651 21450 22256
23031
```

```
23031
[14] 23822 24596 25494 26372 27259 28087 28942
```

By including a "decision rule" in which people drink soda conditional on its popularity, we can simulate how much government intervention would have to change the maximum number of people willing to consume soda (parameter *m*) to affect diabetes rates. In our model, we find that even changes to the number of people willing to consume soda may produce modest changes in diabetes rates, even over very long periods of time. For example, suppose we lowered the parameter *m* from a value of 1 to a value of 0.5 (only half of people are willing to drink soda):

```
> totdm
[1] 14878 15392 15901 16357 16861 17350 17871 18415 18936 19466 19979 20479
20996
[14] 21495 22024 22557 23052 23570 24086 2460
```

By contrast, we can examine how much reducing the social diffusion parameter *q* compares; if we cut *q* in half (from 0.1159 to 0.1159/2), we see that the rates of diabetes are cut to a lesser extent:

```
> totdm
[1] 14819 15324 15866 16392 16919 17496 18032 18578 19145 19743 20376 21000
21619
[14] 22280 22980 23632 24315 24999 25693 26407
```

We could do repeated simulations to compare strategies that affect either or both of these parameters *m* and *q*. The simulation can thereby help us estimate the impact of efforts that reduce the influence of peer pressure, because preventing just one person from drinking soda may have a multiplicative effect when that individual does not subsequently influence others to drink soda. The multiplicative effect of social influences would have been missed by a traditional microsimulation model—highlighting how an agent-based simulation may assist in revealing phenomena that could not be accounted for if we assumed that a population did not change over time in its disease-relevant features, such as personal behavior.

As shown in this example, both the power and pitfall of agent-based modeling is its complexity: agent-based models can potentially incorporate numerous complexities of how people behave, contingent on the unique circumstances of individuals and the people around them. Yet, the "garbage in, garbage out" rule applies—meaning that having good data about people's actual behavioral responses can be critical to implementing an informative agent-based model. Because actual behavioral responses are difficult to identify, researchers often use agent-based modeling for theoretical purposes, such as to identify whether certain kinds of behavioral responses at an individual level produce different results at a population level. For example, in a classical agent-based model of housing segregation in the United States, sociologists found that even if individuals have only slight aversions to living next to people of a different race than themselves, an overall city can become profoundly segregated over time because those small individual biases translate into a very large bias at the population level. This type of finding—in which a population-level phenomena is observed from individual-level behaviors—is known as *emergence*. By revealing emergent phenomena, agent-based models can help to understand why population characteristics (e.g., profound city-level racial segregation) can result from individual characteristics (e.g., only slight racial biases in surveys of racism among individuals) [2].

Another common pitfall of agent-based modeling is to render the behavioral decision rules of simulated people (agents) in the model too complex, such that it is not clear what parameters are producing the overall outcome from the model. For example, our soda consumption model had only two parameters related to the proportion of people willing to consume soda, and the speed at which they are influenced by their peers to start consuming soda. However, several other factors may influence in individual's behavior. To clearly define parameters so that independent researchers can replicate the results of an agent-based model, standard international guidelines have been defined (discussed further in Chapter 11) [3], which indicate that researchers should clarify whether a parameter describes "adaptation" (whether and how an individual in the model responds to changes in their properties or properties of their environment); "fitness" (whether and how an individual explicitly calculates how well they are meeting an objective in order to change their behavior, analogous to the concept of fitness in a Darwinian evolutionary sense); "prediction" (whether and how an individual tries to estimate the future consequences of their decision); "sensing" (whether and how an individual senses changes

in their properties or properties of their environment, such as how many other people around them are drinking soda); and "interaction" (whether and how an individual interacts with others). By precisely defining behavioral decision rules according to these or similar classification systems, we can hope to understand how different ways of conceptualizing behavior (an inherently complex task) can be scientifically studied to explain real-world phenomena including risky health behaviors such as tobacco smoking or soda consumption. As with the code in this textbook, however, the best way to ensure understanding and replication of a model is to share the code producing the model itself.