Chapter 2
Simple Deterministic IPM

Abstract Easterling et al. (2000) originally proposed a size-structured IPM as an alternative to matrix projection models for populations in which demographic rates are primarily influenced by a continuously varying measure of individual size. That model was deterministic and density-independent, analogous to a matrix projection model with a constant matrix. Nothing could be simpler within the realm of IPMs. In this chapter we use that case to introduce the basic concepts underlying IPMs, and to step through the complete process of building and then using an IPM based on your population census data. To illustrate the process of fitting an IPM to population data, we generate artificial data using individual-based models of the monocarpic perennial *Oenothera glazioviana*, and of the Soay sheep (*Ovis aries*) population on St. Kilda (Clutton-Brock and Pemberton 2004).

2.1 The individual-level state variable

In this chapter, we assume that differences between individuals are completely summarized by a single attribute z, which is a continuous variable (e.g., length) rather than discrete (e.g., juvenile versus adult). Ideally z will be the character most strongly linked to individual survival, growth, and reproduction, though z cannot be (for example) an individual's realized growth rate or fecundity. The premise of an IPM is that individuals with the same current state z have the same odds of different future fates and states, but what actually happens involves an element of chance. Individuals within a population typically vary in many different ways. But as a starting point, we assume that to predict an individual's chance of living to next year, how many offspring it will have, etc., it helps to know z but knowing anything more doesn't lead to better predictions. We often refer to z as "size," meaning some continuous measure of body size such as total mass, volume, or log of longest leaf length. But z can be unrelated to size – for example, it could be the individual's spatial location in a linear habitat such as a riverbank, or a bird's first egg-laying date. However, z must

S.P. Ellner et al., *Data-driven Modelling of Structured Populations*,
Lecture Notes on Mathematical Modelling in the Life Sciences,
DOI 10.1007/978-3-319-28893-2_2

have finite limits: a minimum possible value L and a maximum value U. In Chapter 8 we consider models on infinite spatial domains, but those behave very differently from models with finite limits to the value of a trait.

In practice, to predict an individual's fate many possible measures of individual size or state could be used (e.g., the mass or snout-vent length of a crocodile; a plant's total leaf area or the length of its longest leaf; and so on), these may be transformed (log, square root, etc.), and in some cases additional covariates might be used. We cannot emphasize strongly enough the importance of finding the best models to forecast survival, fecundity, and changes in size. This step is like any other statistical analysis of demographic data, so there is no all-purpose recipe, but there are standard tools readily available. For now we assume that you've made the right choices, but we come back to this issue in Section 2.7.

2.2 Key assumptions and model structure

The state of the population at time t is described by the size distribution $n(z,t)$. This is a smooth function of z such that:

$$\begin{gathered}\text{The number of individuals with size } z \text{ in the interval } [a,b] \\ \text{at time } t \text{ is } \int_a^b n(z,t)\,dz.\end{gathered} \tag{2.2.1}$$

A more intuitive description is that

$$\begin{gathered}\text{The number of individuals in the size interval } [z, z+h] \\ \text{at time } t \text{ is approximately } n(z,t)h \text{ when } h \text{ is small;}\end{gathered} \tag{2.2.2}$$

as $h \to 0$, this approximation becomes exact (the relative error goes to zero). Note that $n(z,t)$ is not a probability distribution: its integral over the range of all possible sizes is the total population size, not 1. We will use *probability distribution* or *frequency distribution* to denote a distribution whose integral or sum is necessarily 1.

The model operates in discrete time, going from times t to $t+1$ (the unit of time is often a year, but it can be any duration such as one decade, one day, etc.). Between times t and $t+1$, individuals can potentially die or change in size, and they can produce offspring that vary in size. To describe the net result of these processes we define two functions $P(z',z)$ representing survival followed by possible growth or shrinkage, and $F(z',z)$ representing per-capita production of new recruits. In both of these, z is size at time t and z' is size at time $t+1$. For an individual of size z at time t, $P(z',z)h$ is the probability that the individual is alive at time $t+1$ and its size is in the interval $[z', z'+h]$ (as with $n(z,t)$ this is an approximation that is valid for small h, and the exact probability is given by an integral like equation 2.2.1). Similarly, $F(z',z)h$ is the number of new offspring in the interval $[z', z'+h]$ present at time $t+1$, per size-z individual at time t.

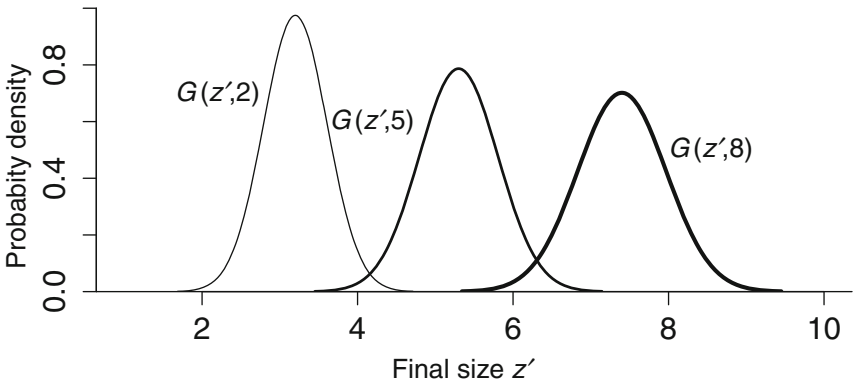

Fig. 2.1 Probability densities $G(z', z)$ of subsequent size z' conditional on survival, for prior sizes $z = 2, 5$, and 8. In this example, z' size has a normal distribution with mean $\bar{y} = 0.8 + 0.7z$ and variance that increases with prior size. Source file: `GrowthDensity.R`

.

It is often convenient to break P up into two processes: survival or death, and size transitions from $z \to z'$:

$$P(z', z) = s(z)G(z', z) \tag{2.2.3}$$

where s is the survival probability, and $G(z', z)$ describes size transitions. In this case we have assumed that both survival and growth depend on initial size, z, as this is often the case, but there are other possibilities, as described in the next section.

With P separated into survival and growth, the kernel $G(z', z)$ can be thought of as a family of univariate probability densities for subsequent size z', that depend on initial size z (see Figure 2.1). In particular, it's always the case that $\int_L^U G(z', z)\, dz' = 1$ for all z; making sure that this is actually true is an important check on how an IPM has been constructed and implemented. If you aren't familiar with probability densities, please read Appendix 2.9. But we recommend that you do this after reading the rest of this chapter, so that you'll have seen some examples of how probabilities densities are used to model population processes.

The net result of survival and reproduction is summarized by the function

$$K(z', z) = P(z', z) + F(z', z) = s(z)G(z', z) + F(z', z) \tag{2.2.4}$$

called the *kernel* - we will refer to $P(z', z)$ and $F(z', z)$ as the survival and reproduction components of the kernel. The population at time $t+1$ is just the sum of the contributions from each individual alive at time t,

$$n(z', t+1) = \int_L^U K(z', z)n(z, t)\, dz. \tag{2.2.5}$$

The kernel K plays the role of the projection matrix in a matrix projection model (MPM), and equation 2.2.5 is the analog for the matrix multiplication that projects the population forward in time.

P and F have to be somewhat smooth functions in order for equation (2.2.5) and the theory developed for it to be valid. We have previously assumed that they are continuous (Ellner and Rees 2006), but it's also sufficient if P and F have some jumps.[1] Consequently an IPM can include piecewise regression models, such as a fecundity model that jumps from zero to positive fecundity once individuals reach a critical "size at maturity" (e.g., Bruno et al. 2011), or a growth model in which individuals cannot shrink.

Looking at equation (2.2.5), it can be tempting to think of $n(z,t)$ as the *number* of size-z individuals at time t, but this is incorrect and will sometimes lead to confusion. It is important to remember the actual meaning of $n(z,t)$ as explained in equations (2.2.1) and (2.2.2). One time when it's essential to keep that in mind is when you want to move from one scale of measurement to another. For example, suppose that you have built an IPM in which z is log-transformed size, and you iterate it to project a future population $n(z,100)$. Now you want to plot the distribution of actual size $u = e^z$; let's call this $\tilde{n}(u,100)$. It's tempting to think that $\tilde{n}(u,100) = n(\log(u),100)$ because having size u is the same as having log-transformed size $z = \log(u)$. But this is wrong, because $n(z,t)$ really isn't the number of size-z individuals. We explain the right way in Appendix 2.9. You should read that section eventually, but that should wait until you've read the rest of this chapter and spent a while working with IPMs on the computer.

2.3 From life cycle to model: specifying a simple IPM

The kernel functions s, G, and F describe all the possible state-transitions in the population, and all possible births of new recruits. But where do these functions come from? Our goal in this section is to answer that question by showing how to translate population census data into a simple deterministic IPM for a size-structured population. We will describe how information on growth, survival, and reproduction is combined to make a kernel. As IPMs are data-driven, our aim is to show how to arrive at a model which is both *consistent* and *feasible*. By consistent, we mean a model that accurately reflects the life cycle and census regime. By feasible, we mean that the model can be parameterized from the available data. In the case studies later in this chapter, we take the next step of fitting specific models to data.

The key idea is that the kernel is built up from functions that describe a step in the life cycle of the species, based on the data about each step. Throughout, we assume that the data were obtained by marking individuals, and following them over their life cycle with evenly spaced censuses at times $t = 0, 1, 2, \cdots$.

[1] Technically, it's sufficient if there is a finite number of curves that divide the square $\mathbf{Z}^2 = \{L \le z', z \le U\}$ into a set of closed regions, and P and F are defined as continuous functions on the interior of each region ("closed" means that each region includes its boundary as well as the interior of the region). The value of K on the dividing curves can be assigned arbitrarily, since this has no effect on the value of (2.2.5).

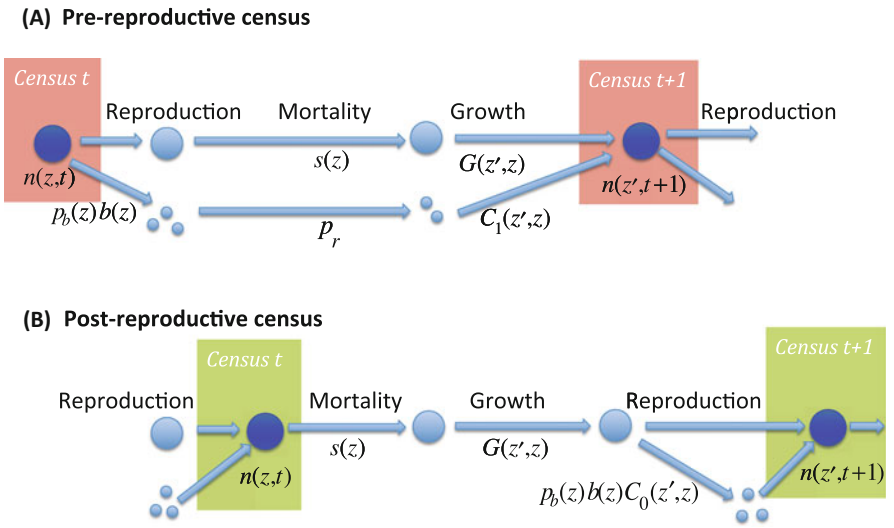

Fig. 2.2 Life cycle diagram and census points for (A) pre-reproductive and (B) post-reproductive census. The sequence of events in the life cycle is the same in both cases. However, the diagrams are different because reproduction splits the population into two groups (those who were present at the last census [large circles], and new recruits who were not present at the last census [small circles]), while at a census time the two groups merge. Reproduction is described by a two stage process with $p_b(z)$ being the probability of reproduction and $b(z)$ being the size-specific fecundity. Each new offspring has a probability p_r of successful recruitment, and its size at the next census is given by $C_1(z', z)$. The pre-reproductive census leads to IPM kernel $K(z', z) = s(z)G(z', z) + p_b(z)b(z)p_rC_1(z', z)$ where $C_1(z', z)$ is the size distribution of new recruits at age 1 (when they are first censused). The post-reproductive census leads to the IPM kernel $K(z', z) = s(z)G(z', z) + s(z)p_b(z)b(z)C_0(z', z)$ where $C_0(z', z)$ is the size distribution of new recruits at age 0 (when they are first censused). The term p_r is absent in the post-reproductive census because new recruits are censused "immediately" after birth, before any mortality occurs.

We strongly recommend that you begin by drawing a life cycle diagram indicating when demographic processes and census points occur, see Figure 2.2. For this first example (Figure 2.2A), we assume that at each time step there is a single census point immediately prior to the next occurrence of reproduction (i.e., there is a *pre-reproductive* census). At that time you record the size of each individual, and we assume that you can distinguish between breeders (those that attempt reproduction) and nonbreeders. At time $t + 1$, the population will include survivors from time t, and new recruits. We assume, for now, that you can assign offspring to parents, and therefore can record how many offspring each individual has.

Combining these data with the size measurements taken at time $t + 1$ gives you a data table like this, suitable for statistical analysis:

Size t	Offspring	Survive	Reproduced	Size $t+1$
3	NA	0	0	NA
7	5	1	1	8
8	4	1	1	10
5	NA	1	0	4

(2.3.1)

Reading across the top row we have the size of an individual in a census, how many offspring it produced (newborns at the next census), whether or not the individual survived to the next census, whether or not it reproduced (this is necessary because an individual might attempt breeding but have no offspring that survive to the next census), and the size of the individual at the next census. Following rows are for other individuals. An NA (indicating "missing data") appears in the Offspring column for any individual that did not attempt reproduction. This is important because it guarantees that only data on breeders will be used to estimate the relationship between size and fecundity. If you cannot distinguish breeders from nonbreeders at census time, the Reproduced column would be absent, the Offspring entry would be zero for all nonbreeders, and fecundity would then be the average over breeders and nonbreeders.

To define the structure of the IPM, we can begin by ignoring individual size and constructing a model for the dynamics of $N(t)$, the total number of individuals at census t, that reflects the life cycle and data. Each individual at time t can contribute to $N(t+1)$ in two ways: survival and reproduction.

- *Survival:* Having observed how many individuals survive from each census to the next, you can estimate an annual survival probability s. At time $t+1$ the population then includes $sN(t)$ survivors from time t.
- *Reproduction:* We could similarly define a per-capita fecundity b, and let $p_r bN(t)$ be the number successful recruits at time $t+1$, with p_r being the probability of successful recruitment. But the data distinguish between "breeders" and "nonbreeders," so we can be more mechanistic. Let p_b denote the probability that an individual reproduces, and b the mean number of offspring produced among individuals that reproduce. Then the number of new recruits at time $t+1$ is $p_b p_r bN(t)$.

Combining survivors and recruits, we have the unstructured model

$$N(t+1) = sN(t) + p_b p_r bN(t) = (s + p_b p_r b)N(t). \tag{2.3.2}$$

The "kernel" for this model is $K = s + p_b p_r b$, which is just a single number because at this point the model ignores the size structure of the population.

The next step is to incorporate how the size at time t affects these rates: the probability of surviving, the probability of reproducing, and the number of offspring are all potentially functions of an individual's current size z:

$$s = s(z), \quad p_b = p_b(z), \quad b = b(z).$$

Our prediction of $N(t)$ can now take account of the current size distribution, $n(z,t)$,

$$N(t+1) = \int_L^U \left(s(z) + p_b(z)p_r b(z)\right) n(z,t)\, dz\ . \tag{2.3.3}$$

At this next level of detail, the kernel is a function of current size z, $K(z) = s(z)+p_b(z)p_r b(z)$. What's missing still from model (2.3.3) is the size-distribution at time $t+1$. To forecast that, we need to specify the size-distributions of survivors and recruits. These are given by the growth kernel for survivors, $G(z',z)$, and the size-distribution of recruits, $C_1(z',z)$, as described in Section 2.2, giving us the complete kernel

$$K(z',z) = s(z)G(z',z) + p_b(z)p_r b(z)C_1(z',z). \tag{2.3.4}$$

for the general IPM (equation 2.2.5).

2.3.1 Changes

We have seen that the structure of the kernel is jointly determined by the life cycle and when the population is censused. To emphasize this essential point, we will now consider how changes in these can lead to changes in the kernel.

Going back to the life cycle diagram (Figure 2.2), what would happen to the structure of the kernel if you had conducted a post-reproductive census? The first thing to notice is that order of events has changed: mortality now occurs *before* reproduction. This has important implications for the structure of the kernel and for the statistical analysis of the data. With a post-reproductive census, the data file will now look like this:

Size t	Offspring	Survive	Reproduced	Size $t+1$
3	NA	0	NA	NA
7	5	1	1	8
8	4	1	1	10
5	NA	1	0	4

(2.3.5)

A crucial difference here is that there are now `NA`'s in both the Offspring and Reproduced column for all individuals that die before the next census. This is because there is no information about what the reproductive success would have been, for individuals who died prior to the next breeding period. As a result, the structure of kernel in this case is

$$K(z',z) = s(z)G(z',z) + s(z)p_b(z)b(z)C_1(z',z). \tag{2.3.6}$$

In order to reproduce, individuals now have to survive, hence $s(z)$ is a factor in both the survival and reproduction components of the kernel. The absence of the p_r term is a consequence of censusing the population immediately after reproduction: there is no mortality between birth and first census. Newly produced individuals do suffer mortality before their next census (at age 1), but this is included in the $s(z)$ term because already at age 0 they are part of $n(z,t)$ (Figure 2.2B). The functions $p_b(z)$ and $b(z)$ are now the probability of reproducing, and mean number of offspring produced, for individuals that survive

the time step. The NAs in the data table make sure that you "remember" this, because they ensure that only data on survivors will be used to fit p_b and b.

When reproduction occurs just before the next census (as in Figure 2.2B), p_b and b could be fitted instead as functions of z', the size at the post-breeding census which is also the size when reproduction occurs. In that approach, the steps to producing size-z' offspring are: survive and grow to some size z^*, breed or not (depending on z^*) and if so have $b(z^*)$ offspring, some of which are size z'. The fecundity kernel needs to total this up over all possible sizes z^*, so we have

$$F(z', z) = s(z) \int_L^U G(z^*, z) p_b(z^*) b(z^*) C_0(z', z^*) dz^*. \tag{2.3.7}$$

Alternatively, the interval between censuses can be broken up into survival and growth in t to $t + \tau$ followed by reproduction in $t + \tau$ to $t + 1$, with separate kernels:

$$\begin{aligned} n(z^*, t + \tau) &= \int_L^U s(z) G(z^*, z) n(z, t) dz \\ n(z', t + 1) &= n(z^*, t + \tau) + \int_L^U p_b(z^*) b(z^*) C_0(z', z^*) n(z^*, t + \tau) dz^* \end{aligned} \tag{2.3.8}$$

We think that equation (2.3.6) is simpler (though you are free to disagree and use equations 2.3.8). When you fit p_b and b as functions of z, you are in effect letting the data do the integrals with respect to z^* for you, because the fitted models will represent the average breeding probability and fecundity with respect to the distribution of possible sizes when reproduction occurs.

The size distribution of new recruits will also vary depending on the timing of the census. In the post-reproductive census, $C_0(z', z)$ is the size distribution of recruits at age 0 immediately after their birth (or so we assume). In the pre-reproductive census, recruits were born immediately after the previous census, so they have already undergone a period of growth before they are first observed, and so $C_1(z', z)$ is the distribution of new recruits aged 1.

Next, let's change the order of events in the life cycle and run through this procedure again, as it's important and potentially confusing. Assume now that death occurs before reproduction – see Figure 2.3. This might describe a temperate deciduous tree or shrub population censused in early spring just before leafout, with growth occurring over the spring and summer and virtually all mortality in winter. Going through the same procedure to construct the kernel, hopefully you can convince yourself that for both the pre- and post-reproductive censuses the structure of the data files and kernels are exactly the same as they were if reproduction occurs before death. The reason for this is that in both cases there is a single census each year, so all demographic rates are based on the size z measured at the start of the time interval. For example in Figure 2.3A,

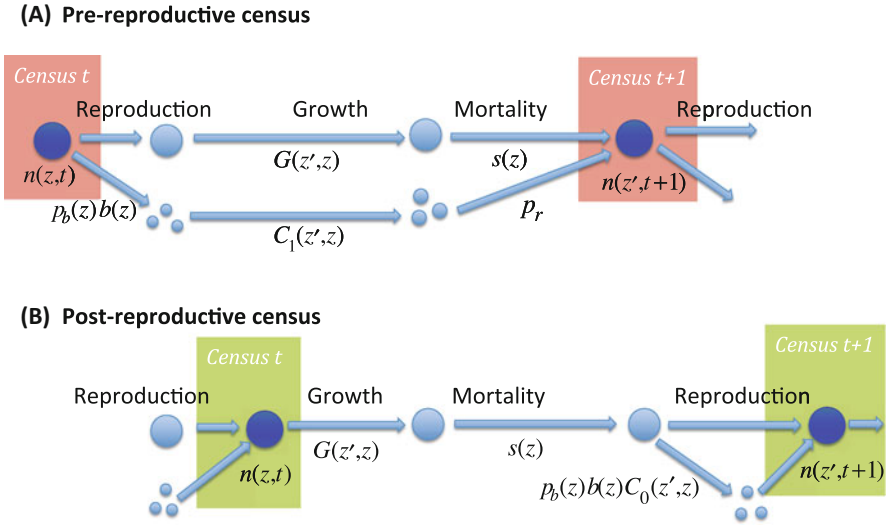

Fig. 2.3 As in Figure 2.2 but with the order of growth and mortality reversed.

the adult survival probability $s(z)$ is based on initial size even though individuals have already grown over the summer, and $s(z)$ then represents the average overwinter survival of trees that were size z at the start of the previous spring.

If there are multiple censuses per year, the additional measurements can be used in constructing the kernel, and the order of events in the life cycle diagram can reflect the timing of events relative to each census. Multiple censuses also mean that we have to decide when we are going to project the population's state. This will typically be one of the census times, but we could also use the average or maximum size over several censuses as our measure of individual size. Alternatively, the annual projection could be broken up into several substeps as in equation 2.3.8, like a seasonal matrix projection model. In Appendix 2.10 we use the Soay sheep case study to present one approach to constructing an IPM when there are several censuses within an annual cycle.

2.4 Numerical implementation

The one-variable IPM (2.2.5) is easy to implement numerically with a numerical integration method called *midpoint rule* (Ellner and Rees 2006). Define *mesh points* z_i by dividing the size range $[L, U]$ into m artificial size classes ("bins") of width $h = (U - L)/m$, and let z_i be the midpoint of the i^{th} size class:

$$z_i = L + (i - 0.5)h, \quad i = 1, 2, \ldots, m. \tag{2.4.1}$$

The midpoint rule approximation to equation (2.2.5) is then

$$n(z_j, t + 1) = h \sum_{i=1}^{m} K(z_j, z_i) n(z_i, t). \tag{2.4.2}$$

A one-variable IPM can always be summarized by the functions $P(z', z)$ and $F(z', z)$. So we assume that you have a script that defines functions to calculate their values:

```
P_z1z <- function(z1,z,m.pars) {
    # your code, for example
    # return( s(z)*G(z1,z) )
    # using functions s and G that you have defined
}

F_z1z<- function(z1,z,m.pars) {
    # your code
}
```

Here `m.pars` is a vector of model parameters. The next step is to compute the mesh points:

```
L <- 1
U <- 10     # size range for the model
m <- 100    # size of the iteration matrix
h <- (U-L)/m
meshpts <- L + (1:m)*h - h/2
```

The R function `outer` makes it easy to compute the iteration matrix:

```
P <- h * (outer(meshpts, meshpts, P_z1z))
F <- h * (outer(meshpts, meshpts, F_z1z))
K <- P + F
```

If you are (like us) always worried that `outer` has not put values in the right places, the next step is to visualize the matrix:

```
matrixImage(meshpts, meshpts, K)
```

The function `matrixImage` which we have written displays a matrix the way they are usually printed (first row at the top, first column at the left). Source `matrixImage.R` and then try it out with:

```
A <- diag(1:10)
A <- matrixImage(A)
```

Box 2.1: Implementing midpoint rule in R

In R (see Box 2.1) we arrange the $hK(z_j, z_i)$ terms in a matrix called the *iteration matrix*. This allows us to iterate the model by matrix multiplication, and use the wide range of numerical tools available for matrices.

The only drawback to midpoint rule is that it's sometimes not very efficient relative to other methods. That becomes an issue when it takes a very large value iteration matrix to get accurate results. In Section 2.7 we give some pointers

on choosing the size of the iteration matrix, and the size limits L, U. In most of this book we'll assume that midpoint rule is efficient enough, but we also discuss alternatives and how to implement them in Section 6.8.

2.5 Case study 1A: A monocarpic perennial

So far we've looked at translating your study system into an IPM, and how to solve the model numerically. In this section we will put all this together to show you that building a basic IPM is really pretty straightforward: there is no black magic or black boxes. To do this we will develop case studies for idealized plant and animal systems, based on published empirical studies. For each, we will simulate data from a individual-based model (IBM) - a simulation that tracks individuals - and analyze the resulting data to build an IPM. To mimic real life, our IBM treats every demographic event as stochastic. Survival versus death is a "coin-toss" (using random numbers) based on a size-dependent survival probability; size at next census for survivors is a random draw from a probability distribution that depends on initial size; and similarly for breeding or not, number of offspring, and so on. All of the individuals in the model follow the same "rules of the game," but apart from that, the model makes them as unpredictable as real plants or animals. The advantage of using simulated data instead of real data is that we know the "truth" and so we can see how various modeling assumptions influence our results, for example what happens if we incorrectly translate the life cycle into an IPM.

2.5.1 Summary of the demography

Monocarpic plants have been extensively modeled using IPMs because of the simplicity of their life cycles, and ease with which the cost of reproduction can be quantified - reproduction is fatal. Our IBM is based on *Oenothera glazioviana*, a monocarpic plant that often occurs in sand dune areas. Its demography has been extensively studied by Kachi and Hirose (Kachi 1983; Kachi and Hirose 1983, 1985). Plant size was measured using rosette diameter - the plants are rosettes of leaves and so can be thought of as disks - and the population censused every month or bimonthly. Plant size is measured in cm and was log-transformed before analysis (see Box 2.2 for an explanation of why we do this). Following Kachi and Hirose (1985) we will use the May census to construct an IBM. This is a pre-reproductive census, so the life cycle is as shown in Figure 2.2A with the additional feature that breeders cannot survive to the next census.

Plants present at the May census flower in July/August (and then die) with probability $\text{logit}(p_b(z)) = -18 + 6.9z$. Those that don't flower survive with probability $\text{logit}(s(z)) = -0.65 + 0.75z$. For those that don't flower and survive, their size next May is given by $z' = 0.96 + 0.59z + \epsilon$, where ϵ has a Gaussian distribution with mean 0 and standard deviation 0.67. Seed production of the flowering plants was estimated in the field, and for a size z plant $b(z) = \exp(1 + 2.2z)$ seeds are produced. The probability a seedling recruits, p_r=0.007, was estimated by dividing the number of seedlings by seed production. Recruits' size at the next census had a Gaussian distribution with mean -0.08 and standard

We are sometimes asked to explain why we often adopt a log transformation of size when building a new IPM. The short answer is that it often works, in that log transformation results in a linear growth model in which the error variance (i.e., the variation in growth) is independent of size. This means that we can model growth using just linear regression rather than more sophisticated methods that use additional parameters to describe the size-variance relationship. And when growth variance still depends on size after log transformation, the dependence is often weak, so that a simple linear or exponential model with just one more parameter is adequate. Another practical advantage of log transformation is that the IPM cannot generate individuals with negative sizes.

The slightly longer answer is that the log transformation makes biological sense when using a linear model to describe growth. For the moment, let x denote some absolute measure of size and $z = \log x$, and assume that growth is completely deterministic. Fitting a linear model using absolute size, $x' = A + Bx$, the expected growth increment $\Delta x = x' - x$ is a strictly decreasing or increasing function of size. That is, $\Delta x = A + (B-1)x$. This is a decreasing function of size if individuals exhibit determinate growth ($B < 1$). However, in many species (e.g., trees) we observe a hump-shaped relationship between size and the absolute growth increment. This is precisely the relationship that emerges if we instead assume that the expected change in log size is a linear function of log size, and therefore fit a linear regression to successive values of log size, $z' = a + bz$. For species with determinate growth ($b < 1$), this implies that the relative growth rate $\log(x') - \log(x) = z' - z = a + (b-1)z$ is a decreasing function of size. Under this model the relationship between the absolute growth increment and size is $\Delta x = e^a x^b - x$. This is hump-shaped when $b < 1$.

Growth is a complex phenomenon, reflecting resource availability, competition, and resource allocation. However, one fairly general explanation for the hump-shaped pattern arises from a consideration of energy acquisition and maintenance costs. All else being equal, larger individuals typically acquire more resources than smaller conspecifics, which means they have more energy available to spend on growth, reproduction, and maintenance. When individuals are small, maintenance costs increase slowly with size relative to acquisition, resulting in a positive relationship between size and absolute growth rate. Later in life when individuals are large, maintenance costs increase more rapidly with size relative to acquisition, leading to a negative relationship between size and growth. Such ideas can be formalized using dynamic energy budget theory.

Ultimately, the growth model should be guided by the data, and should produce good model diagnostics without over-fitting the data. It is also worth keeping in mind that you can fit a demographic model on one scale (e.g., log-transformed size) and building the IPM to work on another. Appendix 2.9 at the end of this chapter explains how to do this.

Box 2.2: On log-transforming individual size.

deviation 0.76. As we don't know which seedling was produced by which parent, the offspring size distribution is a probability density $c_0(z)$ that is independent of parental size z.

2.5.2 Individual-based model (IBM)

With information about the life cycle and the associated census regime in hand, we are now in a position to build an IBM and generate some artificial data for use in constructing our first "data-driven" integral projection model. The IBM is parameterized from the fitted linear and generalized linear models describing survival, growth, and recruitment as functions of individual plant size described in the previous section (Kachi and Hirose 1985; Rees et al. 1999). This ensures that the IBM is realistic, in the sense that the demography is roughly comparable to that of the original *Oenothera* population, but that we still know the true parameters underlying the simulated data.

To make the code simpler to follow we have divided the code into 3 sections, 1) `Monocarp Demog Funs.R` contains the demographic function that describe the how size influences fate, and also the functions used to define and make the iteration kernel; 2) `Monocarp Simulate IBM.R` simulates the individual-based model, and 3) `Monocarp Calculations.R` implements various calculations. If you're not interested in how the data was generated then you can ignore `Monocarp Simulate IBM.R` for now.

In the `Monocarp Demog Funs.R` script we first store the parameters estimated from the field in `m.par.true`, and each is named to make the subsequent formulae easier to interpret.

```
m.par.true <- c(surv.int =  -0.65,
                surv.z   =   0.75,
                flow.int = -18.00,
                flow.z   =   6.9,
                grow.int =   0.96,
                grow.z   =   0.59,
                grow.sd  =   0.67,
                rcsz.int =  -0.08,
                rcsz.sd  =   0.76,
                seed.int =   1.00,
                seed.z   =   2.20,
                p.r      =   0.007)
```

Here `.int` indicates the intercept, `.z` the size slope, and `.sd` the standard deviation, so to access the survival regression intercept we use `m.par.true["surv.int"]`. The various demographic functions are then defined, these are described in detail in Section 2.5.4. Finally we wrapped up the code to make the iteration matrix (Box 2.1) in a function `mk_K` which we pass the number of mesh points `m`, parameter vector `m.par`, and the integration limits `L,U`.

The `Monocarp Simulate IBM.R` script runs the IBM, and for those interested in this we provide in the rest of this subsection a brief overview of what the

code does - this is not essential. We simulate the population as follows. Starting with reproduction, we generate a Bernoulli (0 or 1) random variable, $Repr \sim Bern(p_b(z))$, for each plant to determine which ones reproduce. The function $p_b(z)$ is a size-dependent probability of reproduction estimated from a logistic regression with a logit link function. This means that by applying the inverse of the logit transformation we can write this probability as $p_b(z) = (1 + e^{-\nu_f})^{-1}$, where the estimated linear predictor is $\nu_f = -18 + 6.9z$. Each plant that flowers then produces a Poisson number of seeds, $Seeds \sim Pois(b(z))$, where $b(z)$ is the expected number of seeds predicted from a Poisson regression with a log link function, such that $b(z) = \exp(1 + 2.2z)$. To go from seeds to new recruits, we then simulate *a single binomial random variable* for the whole population, $Recr \sim Bin(S_T, p_r)$, where p_r is the fixed establishment probability (=0.007) and S_T is the total seed production by all flowering plants. To complete the recruitment process we assign a size to the recruited individuals by simulating a Gaussian random variable, $Rcsz \sim Norm(\mu_c, \sigma_c)$, where μ_c (= −0.08) and σ_c (= 0.16) are the mean and standard deviation of new recruit sizes.

Having dealt with flowering and recruitment, we now need to simulate survival and death of established individuals. For each nonflowering plant ($Repr = 0$) we first simulate a Bernoulli random variable, $Surv \sim Bern\,(s(z))$, where $s(z)$ is the size-dependent probability of survival. This was estimated from a logistic regression, and therefore has the form $s(z) = (1 + e^{-\nu_S})^{-1}$, where $\nu_S = -0.65 + 0.75z$. Following the survival phase we allow the surviving ($Surv = 1$) individuals to grow by simulating a Gaussian random variable, $z1 \sim Norm(\mu_G(z), \sigma_G)$, where $\mu_G(z) = 0.96 + 0.59z$ is the expected size of an individual at the next census given their current size and σ_G is the standard deviation (= 0.67) of the conditional size distribution, estimated from a linear regression. Note, the IBM is slightly more complicated than it needs to be as we also track the ages of each individual, as this is needed in the next chapter.

2.5.3 Demographic analysis using lm and glm

The results of the simulation are stored in an R data frame (`sim.data`) and to keep life simple the columns are named in such a way that they correspond to the random variables we used in the IBM. Using the IBM we can generate a data set of any size we want, but to make the example realistic we sampled 1000 individuals from the simulated data. Before we do any analysis we should always check the data file has the right structure, for example flowering plants (`Repr`= 1) need `NA`'s in the `Surv` (survival) and `z1` (next year's size) columns. This is because flowering is the first event in the life cycle and flowering plants die and so they have already been removed from the population before we assume mortality and growth occur. Table 2.1 shows a snippet from the data frame, which has the NAs in the right places.

In order to begin building an IPM we need to fit a series of models capturing the size-dependent rates of survival and reproduction, along with a pair of functions capturing the growth of established individuals and size distribution of recruits. The R code for fitting the required models is fairly simple. Survival is modeled by a logistic regression, so we fit it by

Table 2.1 A few lines from the data frame produced by the monocarp IBM.

z	Repr	Seeds	Surv	z1	age
1.80	0	NA	1	0.92	5
-0.62	0	NA	0	NA	0
0.54	0	NA	1	0.42	1
1.45	0	NA	0	NA	1
-0.12	0	NA	0	NA	0
1.90	0	NA	0	NA	2
3.03	1	2161	NA	NA	1
0.45	0	NA	1	-0.01	1
1.67	0	NA	0	NA	3
-0.37	0	NA	1	0.34	0
0.53	0	NA	1	0.51	0

```
sim.data.noRepr <- subset(sim.data, Repr==0)
mod.Surv <- glm(Surv ~ z, family = binomial, data = sim.data.noRepr)
```

The first line selects the nonreproductive individuals ($Repr = 0$), and then we fit the glm to this subsetted data frame. We remove the flowering individuals because flowering is fatal and occurs before other events in the life cycle. We could have just fitted the glm using `sim.data` as the `NA`'s would have been removed in the analysis, however by explicitly subsetting the data frame it is easier to see what the analysis is doing. The probability of flowering analysis is essentially the same,

```
mod.Repr <- glm(Repr ~ z, family = binomial, data = sim.data)
```

For these plants that flower ($Repr = 1$), we fit the fecundity function as

```
sim.data.Repr <- subset(sim.data, Repr==1)
mod.Seeds <- glm(Seeds ~ z, family=poisson, data=sim.data.Repr)
```

For growth and recruit size we fit the models using linear regression,

```
mod.Grow <- lm(z1 ~ z, data = sim.data)
sim.data.Rec <- subset(sim.data, age==0)
mod.Rcsz <- lm(z ~ 1 , sim.data.Rec)
```

where we use the `NA`s to remove the flowering or dead individuals from the growth analysis, and for recruit size fit a model with just an intercept as parental size doesn't influence recruit size, so recruit size has a Gaussian distribution with constant mean. Finally we estimate the probability of recruitment by dividing the number of recruits by total seed production. The statistical analysis presented here (R code in `Monocarp Calculations.R`) is simplified as we know the form of the functions that generated the data; in a real example we would plot

residuals, test the regression assumptions, and explore a range of alternative models (see Section 2.7), but for now we can cheat a bit. Having fitted the various models we then store the parameter estimates in `m.par.est`, using the same order and names a `m.par.true`.

```
m.par.est <- c(surv    = coef(mod.Surv),
               flow    = coef(mod.Repr),
               grow    = coef(mod.Grow),
               grow.sd = summary(mod.Grow)$sigma,
               rcsz    = coef(mod.Rcsz),
               rcsz.sd = summary(mod.Rcsz)$sigma,
               seed    = coef(mod.Seeds),
               p.r     = p.r.est)

names(m.par.est) <- names(m.par.true)
```

2.5.4 Implementing the IPM

We will use the midpoint rule to implement our IPM and so we need to define the functions $P(z', z)$ and $F(z', z)$, see Box 2.1. The first step is to write down the form of the kernel,

$$K(z', z) = (1 - p_b(z))s(z)G(z', z) + p_b(z)b(z)p_r c_0(z'). \tag{2.5.1}$$

As reproduction occurs first there is no $s(z)$ in the reproduction component of the kernel, and because reproduction is fatal only nonflowering plants survive to next year, hence the initial $(1 - p_b(z))$ in the survival component. The survival component of the kernel, defined in `Monocarp Demog Funs.R`, is given by

```
P_z1z <- function (z1, z, m.par) {
    return((1 - p_bz(z, m.par)) * s_z(z, m.par) * G_z1z(z1, z, m.par))
}
```

For clarity we have written the various functions in the order they occur in the kernel, and the naming of the R functions follows those used in the kernel, equation (2.5.1). For example, survival $s(z)$ is called `s_z`. $s(z)$ was fitted by logistic regression, so the logit of the survival probability is a linear function of the covariates. To calculate $s(z)$ we first calculate the linear predictor using the coefficients of the fitted model stored in `m.par`, and then transform to the probability scale with the inverse logit (i.e., logistic) transformation:

```
s_z <- function(z, m.par) {
    # linear predictor:
    linear.p <- m.par["surv.int"] + m.par["surv.z"] * z
    # inverse logit transformation to probability:
    p <- 1/(1+exp(-linear.p))
    return(p)
}
```

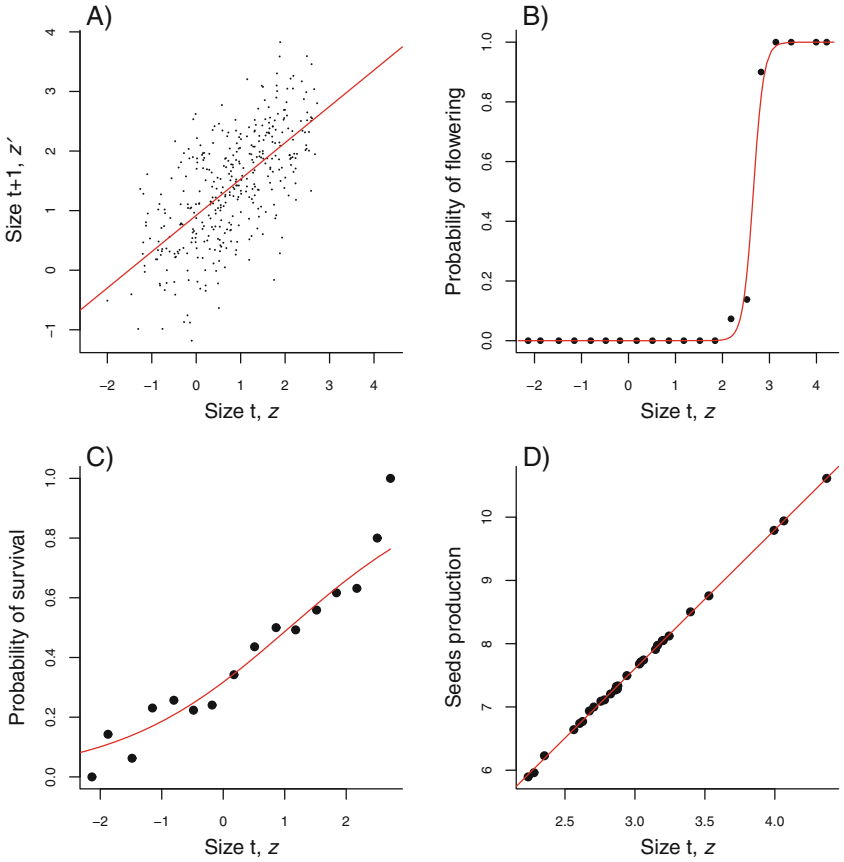

Fig. 2.4 The main demographic processes in the *Oenothera* life cycle. A) Plant size at the next census, B) the probability of flowering, C) the probability of survival, and D) seed production as functions of size at the current census. Source file: `Monocarp Calculations.R`

The $G(z', z)$ function is slightly more complicated as we are going to calculate the probability density function for size at the next census, z' conditional on current size z. To do this we use the regression of z' on z to calculate a plant's expected size at the next census given its current size, which in this case is `mu <- m.par["grow.int"] + m.par["grow.z"] * z`. The standard deviation of z' is determined by the scatter about the regression line in Figure 2.4A, which we also estimate from the fitted regression. Finally because the deviations about the fitted line are assumed to follow a Gaussian distribution, we calculate the probability density function (pdf) of subsequent plant size z' given its current size z using `dnorm`.

```
G_z1z <- function(z1, z, m.par) {
   mu <- m.par["grow.int"] + m.par["grow.z"] * z # mean size next year
   sig <- m.par["grow.sd"]                       # sd about mean
```

```
    p.den.grow <- dnorm(z1, mean = mu, sd = sig) # pdf of new size z1,
                                                 # for current size = z
    return(p.den.grow)
}
```

For the reproduction component of the kernel we have

```
F_z1z <- function (z1, z, m.par) {
  return(p_bz(z,m.par) * b_z(z,m.par) * m.par["p.r"] * c_0z1(z1,m.par))
}
```

As before we use the same naming convention as in equation 2.5.1 for the R functions. With these functions defined and the limits of integration specified we can compute the iteration matrix (see Box 2.1) using the function `mk_K` defined in `Monocarp Demog Funs.R`, as explained below.

2.5.5 Basic analysis: projection and asymptotic behavior

For the *Oenothera* IBM there is no density dependence and so we might expect the population to grow or decline exponentially and the size distribution to converge to some stable distribution, independent of the initial population structure. Let's simulate and see. The result, Figure 2.5, certainly seems to suggest that the total population size grows exponentially and the size structure settles down to a stable distribution. For the true, `m.par.true`, and estimated, `m.par.est`, parameters we can calculate the iteration matrices using `mk_K` and then calculate their dominant eigenvalue using `eigen`, an industrial strength numerical method for calculating eigenvalues and vectors, and compare this with the IBM. The dominant eigenvalue describes the asymptotic rate of population increase, see Section 3.1.1 for more detail. Here's the R-code:

```
> IPM.true <- mk_K(nBigMatrix, m.par.true, -2.65, 4.5)
> IPM.est  <- mk_K(nBigMatrix, m.par.est, -2.65, 4.5)

> Re(eigen(IPM.true$K)$values[1])
[1] 1.059307
> Re(eigen(IPM.est$K)$values[1])
[1] 1.040795

> fit.pop.growth <- lm(log(pop.size.t) ~ c(1:yr))
> exp(coef(fit.pop.growth)[2])
 c(1:yr)
1.072316
```

The function `mk_K` has four arguments: the number of mesh points, the parameter vector, and the two integration limits, which in this case have been set slightly outside the observed size range from the simulation. Using 250 mesh points (`nBigMatrix<-250`) and true parameters we first calculate the iteration matrix and store the result in `IPM.true`, we then do the same with the estimated parameters

and store the result in `IPM.est`. Using `eigen` we then calculate the real part `Re` of the dominant eigenvalue - note `eigen` calculates all the eigenvalues and vectors of the matrix and stores then in decreasing order. For the IBM we estimate the finite rate of increase by regressing `log(pop.size)` against time. There is good agreement between the IBM and the true and estimated IPMs, although even in this ideal case where we know the correct model and have a reasonable sample size (1000 observations) discrepancies of a couple of percent can occur, particularly in high fecundity systems - a single *Oenothera* plant can produce 10,000s of seeds. For the IBM there's not much else we can do except have multiple runs and hope they all show the same pattern. For the IPM the situation is different, as the conditions for stable population growth are well understood (Ellner and Rees 2006). The simplest sufficient condition is that some iterate of the kernel is a strictly positive function; this can be checked in practice by making sure that in some sufficiently high power of the iteration matrix, all entries are > 0 ("high power" here refers to matrix multiplication, e.g., K^3 means `K%*%K%*%K` in R). Other sufficient conditions, which are less easy to check, will be discussed in Chapter 6.

Mean plant size and the mean flowering plant size seem to settle down to some value (Figure 2.5B and C). How can we calculate these from the IPM? We can use `eigen` to calculate the dominant eigenvector, $w(z)$, for the iteration matrix.[2] For IPMs, like matrix models, the dominant right eigenvector is the stable distribution of the individual-level state variable: age, stages, or (in this case) sizes. To calculate average values we need the probability density function (pdf) of size - these have the property that they integrate to 1 which $w(z)$ doesn't. To convert our $w(z)$ to a pdf we just divide by $\int w(z)\, dz$. Average size is then given by

$$\bar{z} = \frac{\int w(z)\, z\, dz}{\int w(z)\, dz}. \tag{2.5.2}$$

In R we can do this as follows:

```
> meshpts <- IPM.true$meshpts
> w.est <- Re(eigen(IPM.est$K)$vectors[,1])
> stable.z.dist.est <- w.est/sum(w.est)
> mean.z.est <- sum(stable.z.dist.est*meshpts)
> mean.z.est
[1] 0.4556423
```

Let's unpack that a bit. First we store the mesh points defined in Box 2.1 in `meshpts`, so the subsequent formulae look a bit neater. Then we use `eigen` to find the dominant eigenvector w of the iteration matrix, and scale w so that it becomes a discrete probability density. The entries in `stable.z.dist.est`

[2] An eigenvector of an IPM is a function of z so some authors call it an eigenfunction; we don't.

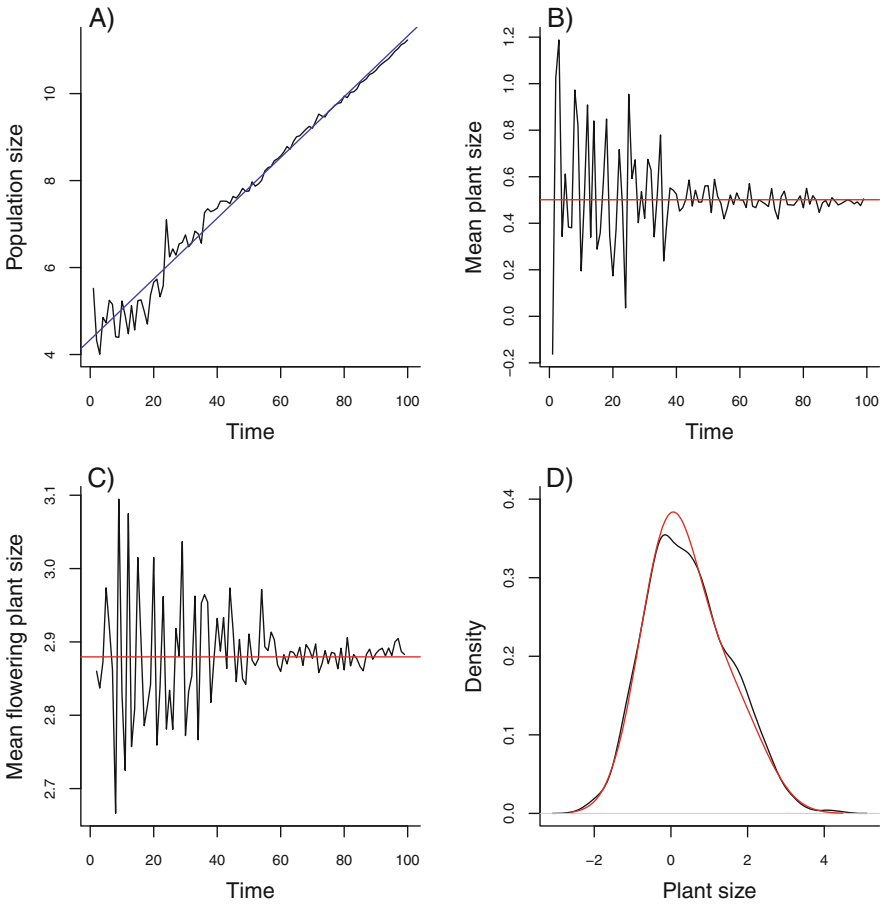

Fig. 2.5 Simulation of the *Oenothera* IBM showing A) log population size, B) mean plant size and C) mean flowering plant size plotted against time. In D) we have plotted the density estimates for size at the end of the simulation. The red lines are calculated quantities from the estimated IPM. Source file: `Monocarp Calculations.R`

then represent the proportion of individuals in each of the size categories $[z_i - h/2, z_i + h/2]$ centered on the mesh points z_i. Finally we compute the mean size by multiplying each z_i by the fraction of individuals whose size is z_i, and summing.

To compute the mean size of flowering plants, we need to take the stable size distribution and weight it by the probability of flowering. The timing of the *Oenothera* census implies that flowering occurs before death, so the stable distribution of flowering plants is $w_b(z) = p_b(z)w(z)$. Then we have to normalize w_b to a probability density, so that it represents the frequency distribution of sizes amongst individuals that flower. The calculations are very similar to those for computing mean size:

```
> wb.est <- p_bz(meshpts,m.par.est)*w.est
> stable.flowering.dist.est <- wb.est/ sum(wb.est)
> mean.flowering.z.est <- sum(stable.flowering.dist.est*meshpts)
> mean.flowering.z.est
[1] 2.985299
```

In each case the calculated means from the fitted IPM provide an excellent description of the simulation data from the IBM (Figure 2.5B and C). If the life cycle and census were such that flowering occurs after mortality we would have to first generate the density of survivors, $s(z)w(z)$, and then multiply by the probability of flowering, so the distribution of flowering plants would be $p_b(z)s(z)w(z)$.

What else might you want to know? The variance of plant size, or of flowering plant size? The fraction of the population that flowers? The effect on population growth of decreasing the flowering probability by 10%? The advantage of using midpoint rule is that intuitive calculations, like those above for mean size, actually work. We *could have* computed means size by approximating the integrals in (2.5.2) using midpoint rule; that would be

```
> stable.flowering.dist.est <- wb.est/ sum(h*wb.est)
> mean.flowering.z.est <- sum(h*stable.flowering.dist.est*meshpts)
```

which gives exactly the same result, as the h's cancel. But we don't have to do that. Instead, we think of the model's state vector $n(z_i, t)$ as a histogram summarizing a size-classified sample from the population, and compute things the way you would compute them from a histogram of data. The results are just as valid as they would be on real data. In this way, any quantity of interest can be computed by iterating the model or by finding its stable distribution, and treating the output as if it was data.

What the IPM leaves out is variation that goes away when you look at the aggregate dynamics of a very large population. Regardless of how large the population is, some kinds of variation don't average out. Consider, for example, a cohort of newborn individuals at time t who are all the same size z. If some individuals of a given size grow, and others shrink, then regardless of how many individuals there are in the cohort, after one time step there will be variation in size. The IPM includes this kind of variation, modeled by the growth kernel $G(z', z)$. The situation is different for *demographic stochasticity*, the random differences in fate among individuals who are separately "playing the same game": two individuals may have (nearly) the same survival probability, but one lives and the other dies: Section 10.3 describes methods for exploring demographic stochasticity in IPMs. In a small population, these chance outcomes can have a big impact. In a large one, however, the the *fraction* of individuals that survives will be less variable, and (in the limit of infinite population size) survival rates equal exactly the size-specific survival probability $s(z)$. You can see this in Figure 2.5A where the deviations from the trend line decrease as the population size increases. This is because the IPM averages out the demographic stochasticity and assumes that exactly $s(z)$ survivors are present next year for each size-z individual present now. So an IPM like the ones in this chapter can't

tell us how the mean flowering size will vary between years as a result of the inherent randomness of mortality. For that you need an individual-based model that really "tosses coins" to decide who lives or dies. Similarly, the IPM does account for variance in recruit size, because with infinitely many recruits, there would still be variance in recruit size. But it does not account for variance in seed production among flowering plants, because with infinitely many parents, the per-capita average fecundity would exactly equal the expected per-capita fecundity.

2.5.6 Always quantify your uncertainty!

Finally, a fitted IPM leaves out the uncertainty that always comes with having a finite sample. If you estimate that $\lambda = 1.1$, what does this mean? You may have been told: it means the population will increase by 10% each year in the long run. But really *it doesn't mean anything*, until you determine how much the estimate of λ is likely to vary from one replicate data set to another.

The simplest and most general way of doing this is by bootstrapping. We will be brief, but only because the topic is covered well by Caswell (2001) and the methods and practical issues are exactly the same for IPMs. It's important, and you should learn how to do it. Quantification of the uncertainty in polar bear population projections (Hunter et al. 2010) was a key ingredient in the decision to list the species as Threatened under the US Endangered Species Act; previous analyses that did not include an uncertainty analysis had resulted in no decision (Hal Caswell, *personal communication*).

The idea behind bootstrapping is to mimic the process by which the sample of observations was drawn from the population, using your data as if they were the full population. Samples generated in this way are used to construct confidence intervals or test hypotheses (Efron and Tibshirani 1993; Davison and Hinkley 1997).

As an illustration, consider a data set of N marked individuals monitored for two years, with information on initial size and on subsequent fate (survival, reproduction, growth, etc.). We can construct an IPM and use this to estimate λ. To quantify the uncertainty in the estimate of λ, we repeatedly draw a sample of size N from the data with replacement. If `X` is a data matrix with row i containing data on the i^{th} individuals, then `Xboot=X[sample(1:nrow(X),replace=TRUE),]` generates a bootstrap dataset. In any such dataset, some individuals occur multiple times and others are left out. Then using each bootstrap dataset in turn, we estimate the IPM parameters and calculate a λ value. The distribution of the estimated λs can then be used to construct confidence intervals (as explained by Efron and Tibshirani 1993; Davison and Hinkley 1997, and implemented in the `boot` package for R, which we recommend).

As an example, the script file `Monocarp Lambda Bootstrap CI.R` uses our monocarp IBM to generate a dataset with 1000 observations, and then bootstrap λ. To generate a bootstrapped distribution of λ the script uses the function `boot`, which has arguments specifying the data and the function to calculate λ; the bootstrapped estimates of λ are stored in the object `boot.out`. There are several

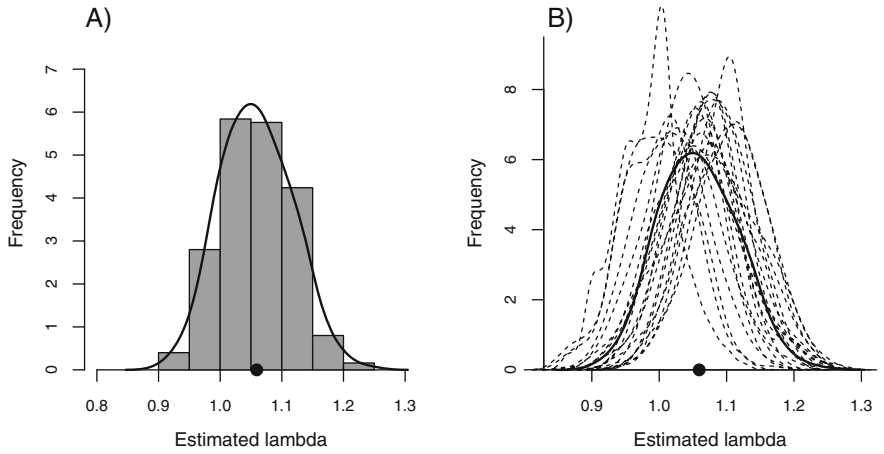

Fig. 2.6 Quantifying the uncertainty in estimates of λ based on a sample of size 1000 (here, coming from simulations of the monocarp IBM). Panel A) shows the distribution of λ estimates across replicate independent samples of size 1000, the histogram and a smooth estimate of the probability density using `density` with `bw.SJ` for the bandwidth. The solid circle at the bottom shows the value of λ for the parameters used to simulate the IPM. Panel B), dashed curves show the probability density of λ estimates from bootstrapping, repeated for 20 different samples. The solid curve is the density from Panel A. Source file `Monocarp Lambda Bootstrap.R`, see also `Monocarp Lambda Bootstrap CI.R`.

ways of calculating bootstrap confidence intervals, and these are implemented in the `boot.ci` function.

```
> boot.ci(boot.out, type = c("norm", "basic","perc"))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = boot.out, type = c("norm", "basic", "perc"))

Intervals :
Level     Normal            Basic             Percentile
95%   ( 0.8817, 1.0679 )  ( 0.8739, 1.0677 )  ( 0.8802, 1.0741 )
Calculations and Intervals on Original Scale
```

Reassuringly the confidence intervals are all similar. Rather less reassuring is the fact that despite having 1000 observations and knowing the correct demographic models the 95% confidence intervals range from ≈ 0.88 to 1.07, so we can't tell if the population is increasing or decreasing; the "true" value of λ was 1.06.

Let's step back a bit and, instead of resampling the data, we use the IBM to generate many replicate datasets. We can then use those to explore how well bootstrapping approximates the variability of estimates across replicate datasets. In Figure 2.6A we've plotted the distribution obtained by repeatedly sampling from the IBM. This defines the ideal, which we are going to approx-

imate by bootstrapping. In Panel B we generated one sample of 1000 plants from the IBM and then bootstrapped λ values from the sample; we repeated this 20 times. From the plot it seems the bootstrapped samples have slightly less variance than the "truth," but in all cases the true value of λ is within the estimated distribution, so the bootstrap distributions have reasonable coverage.

For more complex demographic datasets the bootstrap can be adapted to mimic how the data were sampled (Kalisz and McPeek 1992). For example, if individuals are followed across multiple years, a bootstrapped sample can be constructed by sampling from the list of individuals. Likewise, if the population is stratified by location then bootstrap samples can be drawn from each location and combined. The same is true for sampling designs stratified by time, life stage, genotype, etc.

Confidence intervals on λ can also be constructed by sampling from the posterior distribution of parameter estimates if you've fitted your demographic models in a Bayesian framework; see Metcalf et al. (2009c) for an ecological example involving missing data and density dependence. Within the frequentist setting, we can sample from the estimated distributions of the regression parameters, incorporating the covariance when appropriate, and use these to quantify the uncertainty in the model and its predictions (a good example with explanation of the methods is Pacala et al. 1996). However, because IPMs are built from several separate demographic models, there can be correlations among parameters from different models that use the same data. This is most likely when marked individuals are followed over multiple censuses, so data on one individual may be used in the survival, growth, and fecundity models. An appropriately structured bootstrap accounts for this automatically, but these correlations are omitted when parameters are sampled independently from the estimated parameter distributions of two demographic models.

2.6 Case study 2A: Ungulate

In plant populations, some continuous measure of individual size is often a reliable and easily measured predictor of demographic performance, and key life history transitions such as flowering often depend more on size than on age. An IPM accommodates such demography in a straightforward manner. In contrast, an individual's performance in an animal population is often well predicted by their age or life stage (e.g., mature versus immature), and matrix models yield a good description of the population level processes. Nonetheless, body size or mass is still an key determinant of performance in many animal populations (Sauer and Slade 1987). All else being equal, larger individuals tend to exhibit greater survival and fecundity, so including this state variable can improve projections. In other cases the whole purpose of developing the model is to understand how a continuous state variable impacts demographic processes. For example, IPMs were used to understand how changes in body mass associated with environmental change mediated a major shift in the population dynamics of yellow-bellied marmots, *Marmota flaviventris* (Ozgul et al. 2010). In such cases an IPM is the best tool for the job.

In this section we develop a second case study to illustrate how an IPM can be used to explore body mass-structured dynamics of an archetypal ungulate population. The example is motivated by the well-studied feral Soay sheep (*Ovis aries*) population from the island of Hirta in the St. Kilda archipelago, off the northwest coast of Scotland. We have chosen to base our example on this system because it is one of the richest animal demographic datasets and has consequently been a major target of research into the dynamics and evolution of wild populations. Sheep do not stand still and wait to be counted, but if they're on a small island it's still possible to find them all and weigh them.

In this section our analysis will again be based on a dataset constructed from an individual-based simulation (IBM) parameterized using the Soay dataset. This allows us to simplify some of the details of the life history and keep the case study relatively simple, while again permitting straightforward comparison of the model predictions with "truth." Our aim is to demonstrate that the approach we have sketched out for moving from an individually structured dataset to a fully parameterized IPM applies to almost any life history that can be approximated as a sequence of transitions in discrete time.

2.6.1 Summary of the demography

We will assume that our simulated population is similar to the real Soay sheep population, but with a few important simplifications discussed below. The St. Kilda population has been studied in detail since 1985. Each year newborn individuals are caught, weighed, and tagged shortly after birth in the spring, and body mass measurements are taken from approximately half the population each summer during a catch in August. Maternity is inferred from detailed field observations, while periodic censuses and mortality searches ensure that individual mortality status and population density are very well characterized. Since body mass data on both established individuals and new recruits is only available during the August catch, it makes sense to choose this date as our census point to project the dynamics from. Almost all of the mortality in the system occurs during the winter months when forage availability is low and climate conditions are harsh.

These features of the life history and census regime mean that the life cycle diagram for the Soay system corresponds to the post-reproductive census case in Figure 2.2B, i.e., each annual census occurs after the year's new offspring have been produced but prior to the key mortality period. A potential problem with assuming this sequence of events is that only adults that survive from one summer census until the next should contribute new recruits to the population, despite the fact that lambing occurs several months earlier in the intervening spring. This is demographically equivalent to assuming that any reproducing individual that dies between giving birth in the spring and the summer catch will fail to raise viable offspring. With a handful of exceptions, this is precisely what is observed in the Soay system, so we can consider this to be a reasonable assumption.

To keep our example tractable we make a number of simplifications: (1) we only consider the dynamics of females, that is, we assume the population is demographically controlled by females; (2) we ignore the impact of age-structure; (3) we assume that the environment does not vary among years, either as a result of density-dependence (e.g., resource limitation) or abiotic factors (e.g., winter weather); (4) we assume that Soay females only bear singletons, though in reality they produce twins at a rate 10–15% in any given year. All of these assumptions can be relaxed, although the resulting model is rather more complicated (Childs et al. 2011). Many of these interesting features of the system will be examined in later chapters. Finally, we work with natural log of body mass as the size measure z.

2.6.2 Individual-based model

With the life cycle in mind we can construct an IBM in which individual performance is determined by (log) body mass, z. The IBM was parameterized from a series of linear and generalized linear models fitted to the field data that describe survival, growth, and components of recruitment as functions of individual body mass (DZC, *unpublished analysis*).[3] This ensures that as in the monocarpic plant the demography of the simulated population is similar to the real Soay sheep population, yet we know the true parameters underpinning the data. We adjusted the intercept of survival function so that the population growth rate $\lambda \approx 1.01$.

The IBM is implemented as follows. First, for each individual in the current summer population we simulate a Bernoulli (0-1) random variable, $Surv \sim Bern\,(s(z))$, where $s(z)$ is the size-dependent probability of survival estimated from a logistic regression, and therefore has the form $s(z) = (1+e^{-\nu_S})^{-1}$, where $\nu_S = -9.65+3.77z$. Following the survival phase we allow the surviving individuals to grow by simulating a Gaussian random variable, $z' \sim Norm(\mu_G(z), \sigma_G)$, where $\mu_G(z) = 1.41 + 0.56z$ is the expected mass of an individual next summer given their current size and σ_G is the standard deviation ($= 0.08$) of the conditional size distribution, estimated from a linear regression.

Following survival and growth of established individuals, we simulate a sequence of three processes to add new recruits to the population. For each surviving individual we simulate a Bernoulli random variable which captures reproduction, $Repr \sim Bern\,(p_b(z))$, where $p_b(z)$ is the size-dependent probability of reproduction estimated from a logistic regression. This function is $p_b(z) = (1 + e^{-\nu_b})^{-1}$, where $\nu_b = -7.23 + 2.60z$. Since we assume that a single lamb is born at each reproductive event, the next step is to simulate a Bernoulli random variable, $Recr \sim Bern\,(p_r)$, for each of the reproducing individuals that describes the recruitment of their offspring to the established population next summer. Note, that in this model, the probability of recruitment p_r is independent of parent size. Finally, we assign a mass to the rec-

[3] The data analysis included year-to-year parameter variation; the parameter values used here describe an average year. Likelihood ratio tests were used to assess whether or not keep size in a particular model

ruited individuals ($Recr = 1$) by simulating a Gaussian random variable, $Rcsz \sim Norm(\mu_c(z), \sigma_c)$, where $\mu_c(z) = 0.36 + 0.71z$ is the expected mass of a recruit given their mother's mass, estimated from a linear regression of offspring summer mass against maternal mass *in the previous summer*, and σ_c is the standard deviation ($= 0.16$) of the conditional size distribution.

Starting with an initial population density of 500, we simulated the population until the density reached 5000 individuals and then selected a random sample of 3000 individuals from the simulated dataset to be used in the following analysis. The R code for the demographic functions and the IBM simulation can be found in `Ungulate Demog Funs.R` and `Ungulate Simulate IBM.R`, respectively. The code for running everything and carrying out the analysis we discuss next is in `Ungulate Calculations.R`.

2.6.3 Demographic analysis

The results of the IBM simulation are stored in an R data frame `sim.data` that is essentially identical in structure to the one we used to estimate the parameters of the IBM in the first place. We have named each column so that it matches the definition of the random variables for each life cycle transition described above. We should check this carefully to make sure it has the structure we are expecting. Here is snippet of the data frame:

```
z   Surv   z1   Repr  Recr Rcsz
3.07    0   NA   NA    NA   NA
3.17    1 3.24    1     1 2.47
3.02    1 3.17    0    NA   NA
2.92    0   NA   NA    NA   NA
3.20    1 3.19    1    NA   NA
2.97    1 3.09    1    NA   NA
3.14    1 3.25    0    NA   NA
3.06    1 3.00    0    NA   NA
```

Casual inspection suggests this dataset is in good shape. For example, the one individual that survives ($Surv = 1$) but fails to reproduce ($Repr = 0$) has a sequence of missing values (NA) for the three remaining variables describing offspring recruitment.

Since we know how the data was generated we'll just fit the "right" models to the data and skip the model criticism step. Don't be distracted by the different data frames used to fit each model (e.g., `surf.plot.data` and `repr.plot.data`). We created these from `sim.data` so we use the correct data in each analysis and to help us plot the data and produce summary figures that follow. Survival is modeled by a logistic regression, so we fit it by

```
mod.Surv <- glm(Surv ~ z , family = binomial, data = surv.plot.data)
```

The same is true for whether or not a female reproduced ($Repr$), and whether or not that lamb survived to recruit into the population their first summer ($Recr$),

```
mod.Repr <- glm(Repr ~ z, family = binomial, data = repr.plot.data)
mod.Recr <- glm(Recr ~ 1, family = binomial, data = sim.data)
```

Note that `mod.Recr` does not have any dependence on mother's size z, so it is estimating a single number: the recruitment probability. The `repr.plot.data` data frame was constructed by subsetting the full dataset on the condition that the *Surv* variable equals one, because individuals that do not survive cannot reproduce (survival precedes reproduction). You do not have to do this if you are careful about putting NAs in the right place, but it is good practice as it may help prevent errors cropping up. The subsequent sizes of new recruits and of surviving adults are fitted by linear regression,

```
mod.Grow <- lm(z1 ~ z, data = grow.plot.data)
mod.Rcsz <- lm(Rcsz ~ z, data = rcsz.plot.data)
```

The fitted models are summarized in Figure 2.7. All of the models look fairly reasonable (as they should!). Finally, as in the monocarp example, we extract the parameter values from each of the fitted models and store them in a named parameter vector `m.par.est` that will be used in the IPM script.

2.6.4 Implementing the IPM

The next step is to write down the kernel and check that its formulation matches our knowledge of the life cycle and the data collection protocols,

$$K(z', z) = s(z)G(z', z) + s(z)p_b(z)p_r C_0(z', z)/2 \tag{2.6.1}$$

In this instance both the survival and reproduction kernels contain the $s(z)$ term, because the main period of mortality occurs prior to reproduction (Figure 2.2B). The rest of the survival kernel is just $G(z', z)$ because reproduction is usually not fatal in Soay sheep. The rest of the reproduction kernel is simply the product of the probability of having an offspring, $p_b(z)$, the probability of an offspring surviving to its first census, p_r, and the offspring size distribution, $C_0(z', z)$. The factor of $1/2$ appears because we are tracking only females and assume an equal sex ratio.

This looks like the correct kernel, so we now need to implement the model. We use the approach given in Box 2.1 again so we need to specify the $P(z', z)$ and $F(z', z)$ functions:

```
## Define the survival-growth kernel
P_z1z <- function (z1, z, m.par) {
    return( s_z(z, m.par) * G_z1z(z1, z, m.par) )
}

## Define the reproduction kernel
F_z1z <- function (z1, z, m.par) {
    return( s_z(z, m.par) * pb_z(z, m.par) *
            (1/2) * pr_z(m.par) * C_0z1(z1, z, m.par) )
}
```

There is nothing new here. These are just R translations of the two kernel components, and as before, each function is passed a numeric vector, `m.par`,

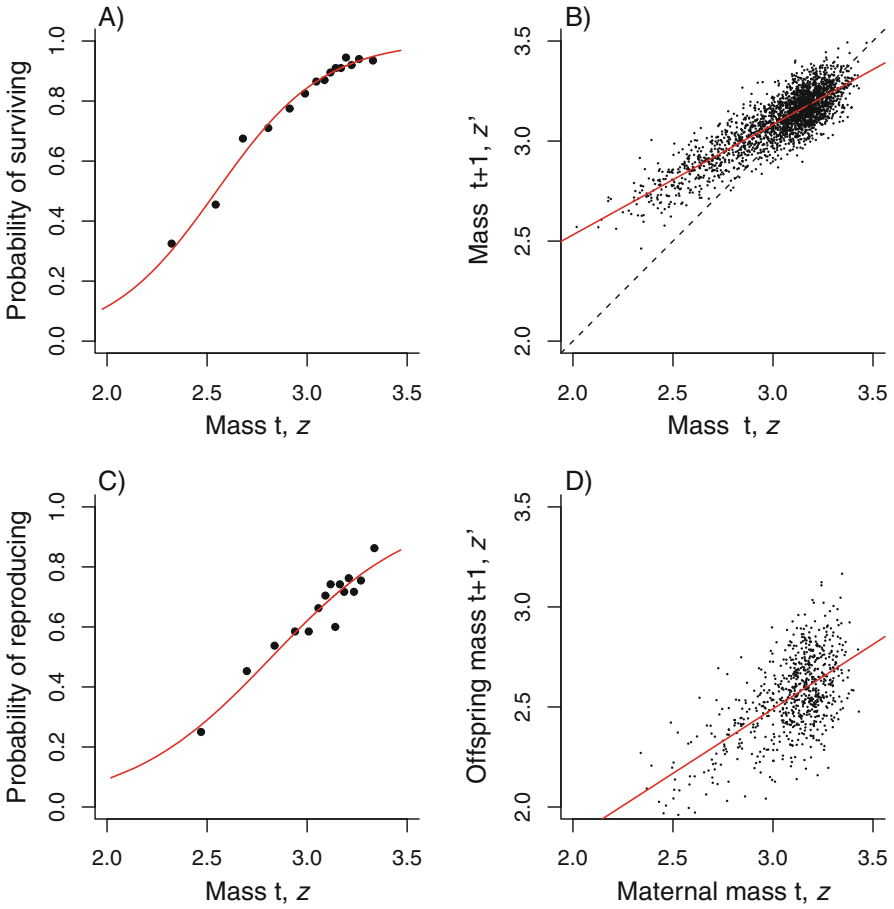

Fig. 2.7 The main mass-dependent demographic processes in the Soay sheep life cycle. A) the probability of survival, B) female mass in the next summer census, C) the probability of reproduction, and D) offspring mass. Source file: `Ungulate Calculations.R`

that holds the parameter values for the underlying demographic regressions. In order to complete the implementation of our model we next need to define the various functions called within `P_z1z` and `F_z1z`. Again, these correspond exactly to the statistical models we fitted to data. For example, the three functions describing the probabilities of survival, reproduction, and recruitment were fitted by logistic regression, as in the *Oenothera* model. So we again obtain the IPM functions by using the estimated coefficients to construct the linear predictor, and the inverse-logit transformation to get the probability:

```
s_z <- function(z, m.par) {
    # linear predictor:
    linear.p <- m.par["surv.int"] + m.par["surv.z"] * z
    # inverse-logit transformation:
```

```
    p <- 1/(1+exp(-linear.p))
    return(p)
}

pb_z <- function(z, m.par) {
    linear.p <- m.par["repr.int"] + m.par["repr.z"] * z
    p <- 1/(1+exp(-linear.p))
    return(p)
}

pr_z <- function(m.par) {
    linear.p <- m.par["recr.int"]
    p <- 1/(1+exp(-linear.p))
    return(p)
}
```

As in the previous example we then need to calculate the probability density function (pdf) for the size at the next census, z' given current size, z, which is done in exactly the same way.

```
G_z1z <- function(z1, z, m.par) {
   mu <- m.par["grow.int"] + m.par["grow.z"] * z # mean size next year
   sig <- m.par["grow.sd"]                       # sd about mean
   p.den.grow <- dnorm(z1, mean = mu, sd = sig)  # pdf for size z1
   return(p.den.grow)
}
```

Finally we calculate the probability density function for recruit size at the next census, given parental size in the current census, using the same approach.

```
> C_0z1z <- function(z1, z, m.par) {
    mu <- m.par["rcsz.int"] + m.par["rcsz.z"] * z # mean size next year
    sig <- m.par["rcsz.sd"]                       # sd about mean
    p.den.rcsz <- dnorm(z1, mean = mu, sd = sig)  # pdf offspring size z1
    return(p.den.rcsz)
}
```

The final step in implementation is choosing the size range and number of mesh points. Because large individuals tend to shrink and their offspring are much smaller than themselves (Figure 2.7), the upper limit just needs to be slightly larger than the largest observed size, and we set $U = 3.55$. The smallest individuals tend to grow, but they have a small chance of breeding and having offspring who are even smaller than themselves. To make sure that the IPM includes those individuals, we can compute the mean offspring size for the smallest observed size ($z \approx 2$) and subtract off two standard deviations of offspring size:

```
> m.par.est["rcsz.int"]+m.par.est["rcsz.z"]*2 - 2*m.par.est["rcsz.sd"];
> rcsz.int
[1] 1.523043
```

So we take $L = 1.5$. The total size range is $U - L \approx 2$ units on log scale. We will use 100 mesh points so that the increment between mesh points is ≈ 0.02 units on log scale, which is about a 2% difference in body mass.

2.6.5 Basic analysis

There is no density dependence in the Soay IBM so we expect the population to grow or shrink exponentially. This is indeed what we find. The finite growth rate of the population (λ) estimated from the simulation is approximately 1.023, so the population is growing by about 2% each year in the IBM. As with the *Oenothera* example (Section 2.5.5), we can estimate the growth rate using the IPM as follows: 1) use the `mk_K` function with the estimated model parameters `m.par.est` to make an iteration matrix; 2) use the `eigen` function to compute the dominant eigenvalue of this matrix, which is our estimate of the population growth rate. When we do this we find that the estimated IPM predicts a lambda of 1.022, which is very close to the value we calculated directly from the IBM output.

In Section 2.5.5 we also showed how to calculate the stable size distribution, $w(z)$, and mean size, $\overline{z}$, using the `eigen` function. This is just a matter of extracting the dominant eigenvector `w`, normalizing it to a discrete probability density function (`stable.z.dist.est <- w/sum(w)`), and using it to compute the mean. What if we want to calculate other central moments of the stable size distribution, such as the variance, σ_z^2? The same logic applies, and with $w(z)$ and $\overline{z}$ in hand such calculations are straightforward. For example, the variance can be written as

$$\sigma_z^2 = E(z^2) - \overline{z}^2 = \frac{\int w(z)\, z^2\, dz}{\int w(z)\, dz} - \overline{z}^2 \tag{2.6.2}$$

where $E(z^2)$ is the expected value of z^2 with respect to the normalized stable size distribution. Assuming that we have stored the stable size distribution (`stable.z.dist.est`) and mean (`mean.z.est`) for the estimated model, the R code for implementing this calculation is

```
> var.z.est <- sum(stable.z.dist.est * meshpts^2) - mean.z.est^2
> var.z.est
[1] 0.07855819
```

That is, we first calculate $E(z^2)$ by multiplying each z_i^2 by the proportion of individuals in the $[z_i - h/2, z_i + h/2]$ size category, and sum these. We then subtract the square of the mean, $\overline{z}^2$, to arrive at the variance. Not surprisingly, this is very close to the size variance estimated directly from the IBM data (=0.078).

More generally, the expected value of any smooth function of size with respect to the stable size distribution can be approximated in this way: first evaluate the function at the meshpoints, then multiply each of these by the corresponding value of the normalized stable size distribution, and sum. For example, if you want to know the mean size of female sheep on the untransformed size scale -

remember, the Soay model works with log body mass - you just need to apply the exponential function to the mesh points first.

```
> mean.z.ari.est <- sum(stable.z.dist.est*exp(meshpts))
> mean.z.ari.est
[1] 20.57083
```

What else might we like to do with the model? In some settings it can be useful to know something about the stable size-age structure implied by the model. For example, knowledge of the implied age structure can provide a useful sanity check; you might worry if the oldest individual ever observed in your study population was 9 years old but your model predicts that 20% of individuals should be older than this. There may be good reasons for such discrepancies (e.g., if the population was recently perturbed) but it they may also indicate that it is time to pause and revisit the model.

The stable size-age distribution can be calculated by using the reproduction and survival kernels to project forward one time-step a cohort derived from a population at the stable size distribution, and separating out the number of newborns, one-year olds, and so on. There is one important snag though: in a growing or shrinking population the relative number of individuals in a given age class are influenced by both the demographic rates and the population growth rate. To begin, we calculate the size distribution of new recruits, $w_0(z)$, using the fecundity component of the kernel and the stable size distribution

$$w_0(z') = \frac{1}{\lambda} \int F(z', z)\, w(z)\, dz \tag{2.6.3}$$

scaling by $1/\lambda$ to compensate for the change in total population. To see why we do this scaling, recall that $\lambda w = (P + F)w$ so that

$$w = \frac{1}{\lambda}(P + F)w.$$

Equation (2.6.3) calculates the part of the w on the left-hand side in the last equation that are newborns, namely $\frac{1}{\lambda}Fw$. Knowing the part of w that are newborns, we can use that in the right-hand side w to calculate the one-year olds as

$$w_1(z') = \frac{1}{\lambda} \int P(z', z)\, w_0(z)\, dz. \tag{2.6.4}$$

The size distribution associated with older individuals can be calculated iteratively in the same way,

$$w_a(z') = \frac{1}{\lambda} \int P(z', z)\, w_{a-1}(z)\, dz. \tag{2.6.5}$$

You can probably guess what the R code associated with these calculations (for the first 4 age classes) looks like.

```
> a0.z.dist.est <- IPM.est$F %*% stable.z.dist.est / lam.est
> a1.z.dist.est <- IPM.est$P %*% a0.z.dist.est / lam.est
```

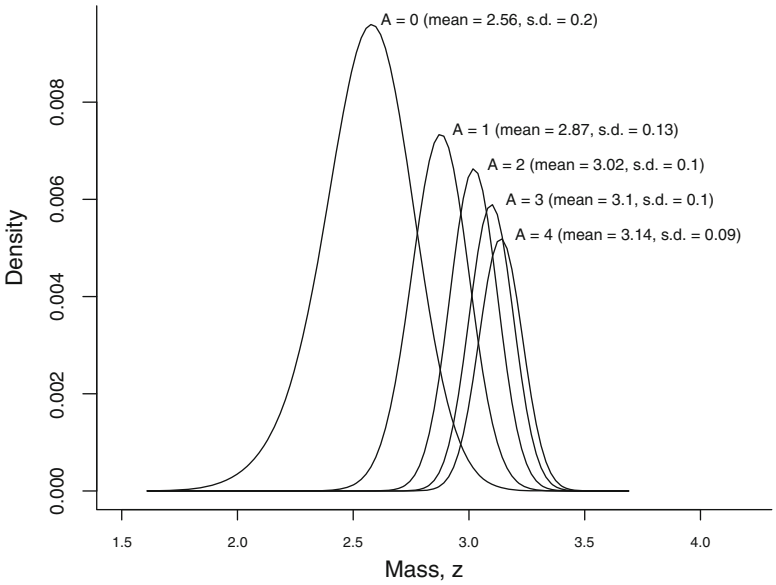

Fig. 2.8 Summary of the stable size-age distribution of the estimated Soay model. Only the first 5 age classes are shown. Source file: `Ungulate Calculations.R`

```
> a2.z.dist.est <- IPM.est$P %*% a1.z.dist.est / lam.est
> a3.z.dist.est <- IPM.est$P %*% a2.z.dist.est / lam.est
> ## and so on...
```

Figure 2.8 plots these distributions to summarize the estimated stable size-age structure of the Soay population.

As in the monocarp case study, the calculations in this section are essentially simulation-based: use the fitted IPM to simulate a population, and then compute statistics from model output in exactly the way you would compute them from data. The calculation of `var.z.est`, for example, is exactly what you would do with a table giving the numbers of individuals in a set of size categories. The calculation of `a3.z.dist.est` and so on corresponds to the experiment: mark a set of individuals at birth, and come back in each successive year to observe how many survived and what their sizes are. Anything you could do with those data can be done in exactly the same way with output from the IPM.

However, many age-related properties (e.g., mean age at first reproduction) and other demographic parameters (e.g., R_0) can be calculated analytically from the IPM kernels and functions. Those methods are the focus of the following chapter.

2.7 Model diagnostics

Each population is a unique situation, so developing a good model is an iterative process of developing candidate models based on the biology and life history of the species, probing those models for faults, and then trying to resolve them.

It's important to double-check a model at all steps, from diagramming the life cycle through implementing the model on the computer.

2.7.1 Model structure

A model such as equation (2.3.4) is a sentence that you can "read out loud" to see if it matches what you believe about the population. The term $s(z)G(z', z)$ says:

> An individual of size z at time t will be size z' at time $t + 1$ if it survives, and then grows (or shrinks) to size z'.

The next term $p_b(z)p_r b(z)C_1(z', z)$ says:

> Production of new offspring is a multistep process. Starting from the current census: an individual has a size-dependent probability of breeding ($p_b(z)$). If the individual breeds, it produces a size-dependent number of offspring ($b(z)$ on average), each of which has probability p_r of surviving to the next census. The size distribution of new recruits that survive ($C_1(z', z)$) is dependent on parent size.

When you read your kernel aloud in this way, it should match your understanding of the species' life cycle.

2.7.2 Demographic rate models

The functions that make up the kernel are statistical models that can be interrogated with standard model diagnostics. Statistical models are often chosen based on tradition, such as logistic regression for survival probability. But tradition is often a reflection of what was computationally feasible half a century ago. Modern computers and R allow us to let the data "speak for themselves" about what models we should fit and let us carefully test the adequacy of simple models.

Figure 2.9 shows a few simple diagnostics for the *Oenothera* growth model $G(z', z)$, again using the artificial data from the IBM held in the data frame `sim.data`. The growth model is a linear regression,

```
mod.grow <- lm(z1 ~ z, data = sim.data)
```

here using a subset of the simulated data so that growth is estimated from observations on about 120 individuals. The first 3 panels are similar to what you would get from R's built-in diagnostics for a linear regression using `plot(mod.grow)`. But we prefer to do it ourselves, so that we can use some features from other packages.

- Residuals should have constant variance and no trend in mean; plotting residuals versus fitted values (panel A) provides a visual check on these properties. The plotted curve is a fitted nonparametric spline curve, using the `gam` function in the `mgcv` package (Wood 2011). It hints at the possibility of a small nonlinear trend, but this may be driven by a few points at the left (and since the data come from the IBM, we know that the underlying growth model really is linear).

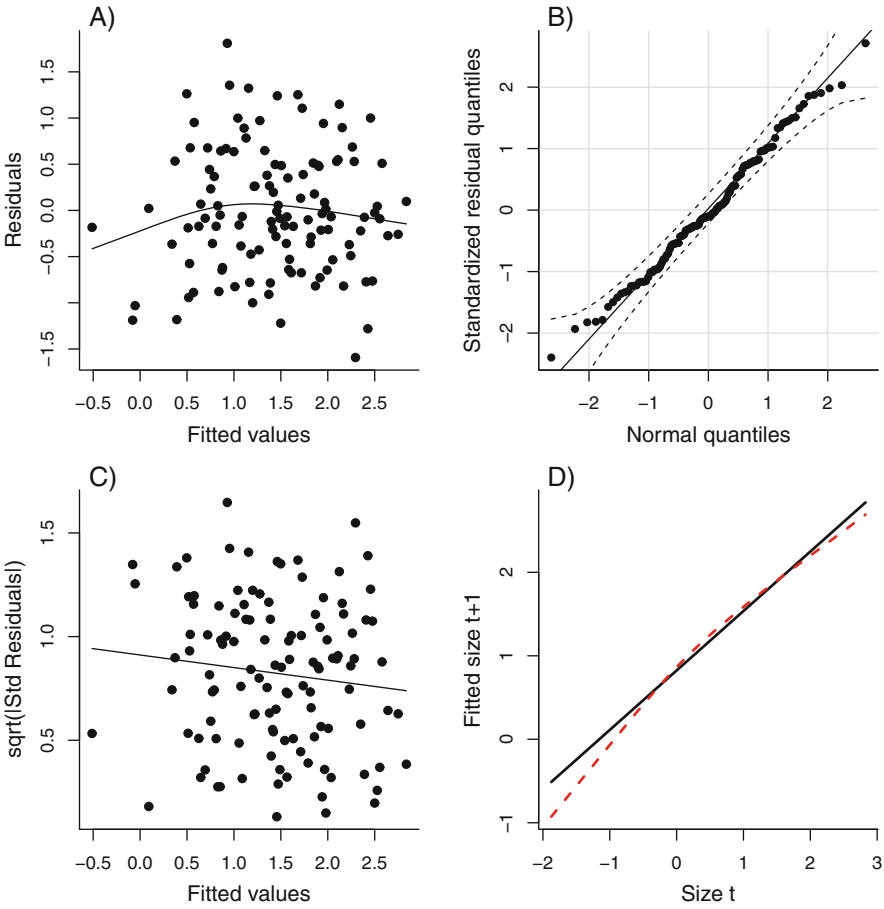

Fig. 2.9 Diagnostic plots for the monocarp growth function; see text for details. Source file: `Diagnose Monocarp Growth Kernel.R`

- Residuals should be Gaussian. A quantile-quantile plot (panel B), using the `qqPlot` function in the `car` library, supports this. Perfectly Gaussian residuals would fall on the 1:1 line (solid). The dashed lines are a 95% pointwise confidence envelope, so we should worry if more than 5% of points fall outside the envelope or if any points lie far outside it. This one looks OK. As a further check, we can test for statistically significant departures from Gaussian distribution:

```
> sresid <- rstandard(mod.grow); shapiro.test(sresid);
Shapiro-Wilk normality test
data:  sresid
W = 0.9909, p-value = 0.6288
```

This confirms what we know: the growth distribution is Gaussian.

- A better check for constant error variance is a scale-location plot (panel C). The plotted points are the square-root magnitude of the standardized residuals, and the curve is again a fitted spline. The spline hints at a possible weak trend so we test for significance (and find none):

```
> cor.test(zhat,sqrt(abs(sresid)),method="k");
Kendall's rank correlation tau
data:  zhat and sqrt(abs(sresid))
z = -1.1606, p-value = 0.2458
```

  "Standardized" means that residuals have been adjusted for differences in leverage between points, which cause residual variance to be nonconstant even when the true error variance is constant. In large data sets standardization usually has little effect because no point has much leverage, but in smaller data sets standardization can remove spurious patterns in the magnitude of residuals.[4]
  In a typical regression analysis the main goal is to estimate the trend represented by the regression line. It's OK if the residuals are "close enough" to Gaussian and the variance is "close enough" to constant. But in an IPM, the scatter around the mean growth rate is also an important part of the model. A smaller growth variance in large individuals might be important for predicting longevity because it keeps big individuals from shrinking, even if it is inconsequential for estimating the effect of size on mean growth rate.
- To follow up on the hint of nonlinearity in panel (A), we can compare the linear model with a spline fit to the same data (panel D, solid and dashed lines).

The spline in panel (D) suggests that growth is a weakly nonlinear function of size. We know this isn't true in this case – it's an accident of random sampling – but with real data we would have to decide between the linear and nonlinear models. The statistical evidence is equivocal: a significance test using `anova(mod.grow,gam.grow)` is marginally nonsignificant ($P = 0.063$), while AIC slightly favors the nonlinear model (AIC=243.3) over the linear model (AIC=244.9).

The AIC difference is small enough that most users would probably select the simpler model. However, its lower AIC means that the nonlinear model is expected to make more accurate predictions (recall that in the frequentist framework AIC is a large-sample approximation to out-of-sample prediction error, which is why frequentists, and agnostics like us, guiltlessly use both AIC and p-values). Moreover, Dahlgren et al. (2011) have shown that even weak nonlinearities can sometimes have substantial effects on model predictions, so the nonlinear model should be taken seriously. When the statistical evidence for one model over another is equivocal, unless one of the models is strongly favored based on some underlying biological hypothesis, we believe that the

[4] What we call standardized residuals are sometimes called Studentized residuals. We follow the terminology used in R.

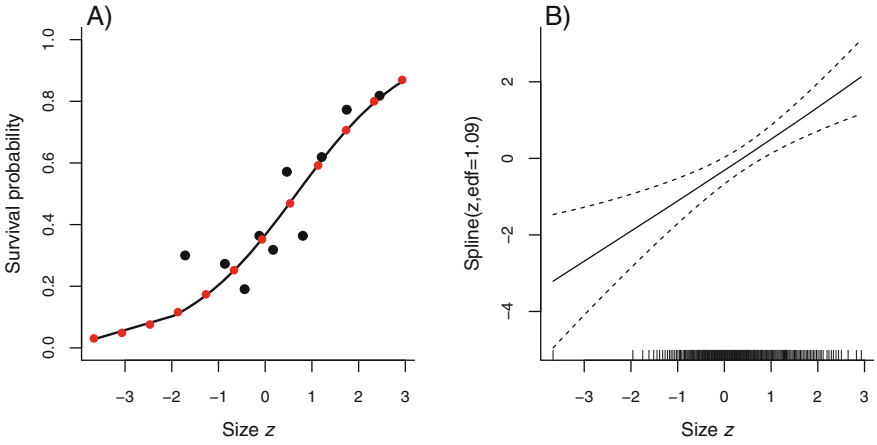

Fig. 2.10 Diagnostic plots for the fitted *Oenothera* survival function. (A) Survival as a function of size. The solid curve is the prediction of the fitted GLM (logistic regression); the red circles are the predictions of the fitted nonlinear GAM (nonparametric logistic regression); the black circles are survival fraction as a function of mean size for a series of size classes defined by percentiles of the size distribution. (B) The result of plotting the fitted GAM using `plot(gam.surv)`, which shows the fitted spline regression (solid curve) on the scale of the regression model's "linear predictor." If this curve is a straight line (with 1 degree of freedom specifying the slope), the GAM is equivalent to the GLM. In this case the fitted GAM has 1.09 "effective degrees of freedom," and the linear model is well within the confidence bands on the GAM estimate. Source file: `Diagnose Monocarp Survival.R`

best approach is to try both models and attach most confidence to conclusions that the models agree on. Model averaging is another possibility. But current model averaging approaches are not always effective (Richards et al. 2011), and we think that it is more informative to show the degree of uncertainty by presenting results from the range of plausible models.

Residual plots are less informative for the survival or flowering models, because all observed values are either 0 or 1 and there is no expectation of Gaussian residuals. One visual check is to compare model predictions with survival rates within size classes (Figure 2.10A). And we can again compare the linear model with a nonlinear model using `gam` (Figure 2.10B). In this case the linear model is supported: it has lower AIC, and the difference between the linear and nonlinear models is minuscule. Further checks are to test for overdispersion, to test whether the fit is significantly improved by adding predictors other than size (e.g., individual age if it is known), and so on.

2.7.3 Implementation: choosing the size range

It seems natural that a model's size range should correspond to the range of observed sizes, perhaps extended a bit at both ends. Many published studies have used this approach. For *Oenothera*, we presented above a "data" analysis based on a sample of 1000 individuals. In a sample of 1000 individuals from the

stable size distribution, the range of observed sizes is typically -2.55 to 3.9 (these were the median min and max sizes from 10000 replicate samples). Because size is on a natural-log scale, setting $[L, U] = [-2.65, 4.0]$ allows the IPM to include individuals ≈10% smaller or larger than any in the sample.

However, it's important to check whether a size range chosen in this sort of way is really big enough that individuals aren't getting "evicted" from the model. *Eviction* refers to situations where the distribution of subsequent size z' extends beyond $[L, U]$, so that individuals in the tails of the growth or recruit size distributions aren't accounted for when the model is iterated (Williams et al. 2012).

Preventing eviction of new recruits is easy: make sure that $[L, U]$ is wide enough to include effectively all of the recruit size distribution. The fitted recruit size distribution is Gaussian with mean of ≈ -0.1 and standard deviation of ≈ 0.8. Some recruits are evicted because a Gaussian has infinite tails. But we can check that this number is small enough to ignore by computing the non-evicted fraction:

```
> pnorm(4,mean=-0.1,sd=0.8) - pnorm(-2.65,mean=-0.1,sd=0.8);
[1] 0.9992823
```

This is the fraction of recruits whose size is below 4, but not below -2.65. So fewer than one recruit in a thousand gets evicted – not a problem.

For older individuals, we need to look at the growth kernel $G(z', z)$. Figure 2.11A shows that with $[L, U] = [-2.65, 4]$ in the *Oenothera* IPM, the largest individuals lose a substantial part of their subsequent size distribution to eviction. So eviction is happening, and the next question is: does it matter? A large *Oenothera* is very likely to flower and die rather than continuing to grow, so what would happen if it lived is irrelevant, both in the model and in reality. To see if this argument really holds up, we can look at the fraction of individuals that the IPM sends to the "right place." For a real *Oenothera* there are three possible fates: flower and die, die without flowering, or survive with some new size z'. In the model there's a fourth possibility: eviction. The probability of going to the wrong place (eviction) for a size-z individual is the integral of $s(z)(1 - p_b(z))G(z', z)$ from $z' = U$ to ∞. Plotting this probability when $U = 4$ (solid curve in Figure 2.11B) turns up a surprise: eviction is most likely in the middle of the size range, where flowering is not a near-certainty. Still, the maximum risk of eviction is under 1%, and relative to other flaws in a data-driven model this one is probably not worth fretting over.

But what if you're not so lucky, and your IPM is sending too many individuals to the wrong place? Or if you're worried that eviction might still affect evolutionary analyses, because it causes reproductive value to be under-estimated?

A useful first step is to ask if eviction is a symptom of flaws in the growth model. Perhaps the data are better described by a different growth model with lower or no eviction: a model with nonconstant growth variance, or a bounded growth distribution (e.g., a shifted and stretched beta distribution). Or perhaps there's some reason why real individuals die before reaching their maximum potential size, and you need to make that happen in the model.

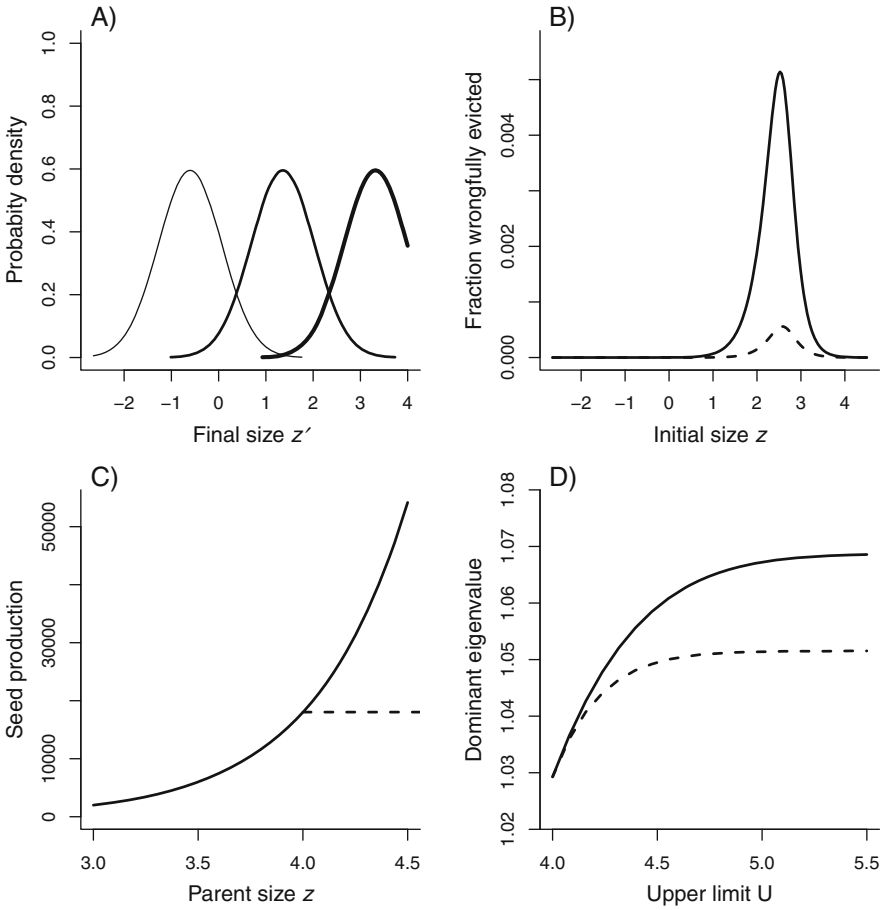

Fig. 2.11 Detecting and eliminating eviction of large individuals in the *Oenothera* IPM. A) Plots of the growth kernel $G(z', z)$ for z at the lower endpoint, midpoint, and upper endpoint of the size range $[L, U] = [-2.65, 4]$. B) Size-dependent probability of eviction with $L = -2.65$ and upper size limit $U = 4$ (solid) and $U = 4.5$ (dashed). C) Per-capita seed production as a function of parent size with no demographic ceiling (solid) and a ceiling at $U_1 = 4$ (dashed). D) Dominant eigenvalue λ as a function of the upper limit U for $L = -2.65$ with no demographic ceiling (solid) and a ceiling at $U_1 = 4$ (dashed). Source file: `MonocarpGrowthEviction.R`

Again, what if you're not so lucky: the growth model fits the data well, but eviction still happens. Williams et al. (2012) present several possible solutions, but caution that the best approach is case-dependent: there's no universally right cure for eviction. In *Oenothera*, as in many other cases, individuals near the top of the size range tend to shrink, so setting $U = 4.5$ reduces the maximum eviction risk to under one in a thousand (Figure 2.11B, dashed curve). However, the model then extends far beyond the largest individual likely to be sampled, and that kind of extrapolation is dangerous. Increasing U from 4 to 4.5 is

almost a doubling of the maximum size, and the maximum fecundity is tripled (Figure 2.11C, solid curve). To avoid creating such Titans, you can impose a demographic ceiling: a size U_1 such that individuals larger than U_1 have the same vital rates as individuals of size U_1. This approach, used by Easterling et al. (2000), is equivalent to adding a discrete class of "large" individuals (all those above the ceiling) who are demographically identical. The dashed line in Figure 2.11C shows per-capita seed production with a demographic ceiling at $U_1 = 4$.

In practice adding a demographic ceiling can be done very simply: a line like

```
K=h*outer(meshpts,meshpts,kernel.function,m.par)
```

becomes

```
K=h*outer(meshpts,pmin(meshpts,U1),kernel.function,m.par)
```

However, if you add a ceiling we recommend making its value one of the entries in the parameter vector `m.par`, and using it in each of the kernel's component functions, so that you can easily explore how sensitive your model's predictions are to the location of the ceiling. Imposing a ceiling is a modeling decision that should be guided by your substantive knowledge: what is the most reasonable way to extrapolate beyond the range of the data? Because the biggest are often the most fecund, how you extrapolate can matter even if the biggest are rare. Figure 2.11D shows how the dominant eigenvalue is affected by the upper size limit with and without a demographic ceiling at $U = 4$. With the ceiling, there's no meaningful gain from increasing the upper limit past $U = 4.5$, but that depends on what has been assumed about fecundity at unsampled sizes. As always, it's important to consider alternative plausible assumptions and how those affect your conclusions.

2.7.4 Implementation: the number of mesh points

The number of mesh points m required for accurate results depends on the kernel. An integral computed by midpoint rule is exactly right if the function being integrated is linear on each of the size classes of width $h = (U - L)/m$ centered at the mesh point. It will be close to right if the function is close to linear on intervals of length h.

For IPMs this means that h needs to be small compared to the standard deviation of offspring size, and to the standard deviation of growth increments $z' - z$. If neither of these is too small, then a simple but effective approach is to start with a reasonable value of m and increase it until numerical results stabilize to the accuracy you need. But if either of these is small, direct application of midpoint rule might lead to a very large matrix and to calculations that take a long time or crash when memory runs out.

There are a several possible ways around this problem. If the offspring size distribution is the problem and the variance of growth increments is much larger, the offspring size variance can be increased without substantially changing model outputs, because the variance in initial offspring size is swamped by

the variance in their first growth increment. Suppose new offspring have standard deviation σ_0 and first year growth has standard deviation $\sigma_G \gg \sigma_0$. Their size at age 1 then has variance $\sigma_0^2 + \sigma_G^2$. Increasing σ_0 to a larger value $\tilde{\sigma}_0$ such that $\tilde{\sigma}_0^2$ is still $\ll \sigma_G^2$ will have a trivial effect on the variance in their size at age 1. Again, there is a simple numerical check on this. Find a value of $\tilde{\sigma}_0$ large enough that model predictions stabilize with increasing m while m is still small enough for your computer to handle. Then increase $\tilde{\sigma}_0$ to $2\tilde{\sigma}_0$, and if the change in model predictions is very small then you're safe. This approach not likely to fail unless small size differences at birth are amplified by a largely deterministic pattern of growth, rather than washed out by variance in later growth increments.

Small variance in growth increments is a more difficult problem. In practice it arises for long-lived, slow-growing species. If growth increments are small relative to the range of individual sizes in the population, the standard deviation of growth increments is necessarily small too. The kernel can't be "fixed" in that case without changing model forecasts. One option then is to use computational methods that can deal with large matrices. For example, the dominant eigenvalue and corresponding eigenvectors can be found by iteration (Ellner and Rees 2006) instead of using `eigen` to compute all of the eigenvalues and eigenvectors. A second option is to use more efficient ways of evaluating integrals, which we discuss in Section 6.8. The third, and likely best, option is to use methods for sparse matrices. These methods (such as those in R's `spam` and `Matrix` libraries) only store and work with the nonzero entries in a matrix. If growth is nearly deterministic, the IPM kernel is nearly zero in large regions representing impossible size transitions. In that case, if you zero out the corresponding entries in the iteration matrix (e.g., set to exactly zero all matrix entries below some threshold, such that the sum of each column is reduced by less than 0.1% of its total), the resulting matrix will be sparse and computations will be much faster with sparse matrix methods. Sparse matrix calculations can also be useful when individuals are cross-classified by multiple traits (Ellner and Rees 2006). We return to these issues and illustrate the methods in detail in Section 6.6.

2.8 Looking ahead

This chapter has covered the tools for building a basic size-structured IPM from data, make projections with it, and ask some probing questions about whether the model is really doing what you want it to do. But there's much more to do than projection. An IPM is not just a population model, it's a model for the lives of individuals, and we can extract a lot of information about individual life histories and their variation. We take this up in the next chapter.

2.9 Appendix: Probability densities versus probabilities, and moving from one scale of measurement to another

One important difference between matrix and integral projection models is that demographic transitions in the matrix model are described by *probabilities*, where an IPM uses a *probability density* for the same transition. For example,

in a size-structured matrix model, the projection matrix entry a_{32} representing transitions from size-class 2 to size-class 3 is the product of the survival probability s_2 for size-class 2, with the probability g_{32} that a survivor grows into size-class 3: $a_{32} = s_2 g_{32}$. In the basic size-structured IPM, we typically have instead

$$P(z', z) = s(z)G(z', z).$$

Here $s(z)$ is the size-dependent survival probability, just like in a matrix projection model. But $G(z', z)$ is the *probability density* of size z' at the next census for survivors having initial size z.

Probability densities are enough like probabilities that you can often ignore the difference. But sometimes you can't. In past writings we have sometimes glossed over the difference, for example calling $G(z', z)$ the probability of growing to size z'. But conversations and emails with IPM builders have convinced us that this shortcut created *much* more confusion than it avoided, so in this book we try to be more accurate.

The density of water is the mass per unit volume of water, such as $\mathrm{g/cm^3}$. To get the mass, you multiply density by volume. Similarly, to get the probability for a given range of body sizes, we have to take the probability density and multiply it by the "volume" of the size range. Because size is one-dimensional, the volume of a size range is its length. For a size range of length h we therefore have

$$\text{Probability that new size } z' \text{ is in } [z_1, z_1 + h] \approx G(z_1, z)h. \tag{2.9.1}$$

Why do we have $\approx$ rather than $=$ in equation (2.9.1)? Why is it just an approximation? The reason is that the probability density for z' is generally not constant, so the density at z_1 doesn't apply over the whole size range $[z_1, z_1 + h]$. To be exact, we have to use the right density for each size, which is accomplished by integration:

$$\text{Probability that new size } z' \text{ is in } [z_1, z_1 + h] = \int_{z_1}^{z_1+h} G(z', z)\, dz. \tag{2.9.2}$$

However, one way you *won't* go wrong is by thinking of equation (2.9.1) as being exactly right, so long as you remember that it only holds for small h (narrow size ranges).

From equation (2.9.1) we see one way in which probability densities are like probabilities: they give us the *relative* likelihood of different outcomes. In this case,

$$\frac{\text{Probability that new size } z' \text{ is near } z_1}{\text{Probability that new size } z' \text{ is near } z_2} \approx \frac{G(z_1, z)h}{G(z_2, z)h} = \frac{G(z_1, z)}{G(z_2, z)} \tag{2.9.3}$$

so long as the same definition of "near" is used at z_1 and z_2. This is the best way to think about the intuitive meaning of probability density: it tells us the

relative probability of two equal-length (small) ranges of the variable, but not the absolute probability of any particular value.

In the same way, regardless of how often we or anybody else has said otherwise, the fecundity kernel $F(z', z)$ is *not* the number of size-z' offspring produced by a size-z parent. Rather, as in equation (2.9.1), $F(z', z)h$ is (for small h) the number of offspring in the size range $[z', z'+h]$ produced by a size-z parent, and $F(z_2, z)/F(z_1, z)$ is the relative frequency of offspring with sizes near z_2 and z_1.

The difference between probability and probability density is crucial when you need to move a kernel from one scale of measurement to another.

Suppose that your data lead you to fit a growth model using a linear ("untransformed") size measure u, but for everything else log-transformed size works better, so you decide to use $z = \log(u)$ as your state variable. Then you need to take the growth model $G(u', u)$ and express it as a growth model on log-scale, $\tilde{G}(z', z)$.

If we think of $G(u', u)$ as the probability of growing to size u', the answer is easy. Going from log-size z to log-size z' is the same as going from size $u = e^z$ to size $u' = e^{z'}$. So this misguided thinking leads us to

$$\tilde{G}(z', z) = G(e^{z'}, e^z) \qquad \longleftarrow \textbf{REMEMBER: this is wrong!}$$

The misguided approach was right in one respect: we need to find $\tilde{G}$ by computing the same probability two different ways. But the right way to compute probabilities is by using equation (2.9.1), which really does involve probabilities. To simplify the calculations we can assume that h is small, and drop terms of order h^2 or smaller.

- $\tilde{G}(z', z)h$ is the probability that subsequent log size z' is in $[z', z' + h]$, starting from log size z.
- In terms of untransformed size $u = e^z$, this is the probability that subsequent size u' is in the interval $[e^{z'}, e^{(z'+h)}]$, starting from size $z = e^u$. For small h, the Taylor series for the exponential function ($e^x = 1+x+\frac{x^2}{2}+\cdots$) implies that $e^h \approx 1 + h$. Thus

$$[e^{z'}, e^{(z'+h)}] = [e^{z'}, e^{z'}e^h] \approx [e^{z'}, e^{z'}(1 + h)] = [e^{z'}, e^{z'} + e^{z'}h],$$

  a size range of length $e^{z'}h$, whose probability is therefore

$$G(e^{z'}, e^z) \times e^{z'}h.$$

We therefore have

$$G(e^{z'}, e^z)e^{z'}h = \tilde{G}(z', z)h$$

and therefore

$$\tilde{G}(z', z) = e^{z'}G(e^{z'}, e^z). \tag{2.9.4}$$

The general recipe is as follows. Let f be the function that gives u (the scale on which G or some other kernel was fitted) as a function z (the scale where the IPM works). In the example above, $z = \log(u)$ so $u = e^z$ and the function f is $f(z) = e^z$. Then the kernel on the scale of the IPM is

$$\tilde{G}(z', z) = f'(z')G(f(z'), f(z)) \qquad \longleftarrow \textbf{This one is right!} \qquad (2.9.5)$$

where $f' = df/dz$. To implement this in an IPM, we recommend writing a function that computes $G(u', u)$ on the scale where the kernel was fitted, and a second function that computes $\tilde{G}$ by using equation (2.9.5) and calling the function that computes G.

Equation (2.9.5) applies also to functions of a single variable. The situation is especially simple for a function of just initial size z: $\tilde{s}(z) = s(f(z))$. For a function of just z' such as a parent-independent offspring size distribution, $\tilde{C}(z') = f'(z')C(f(z'))$.

The same approach works for moving the size distribution from one scale to another. At the end of Section 2.2 we gave the example of using an IPM to project $n(z, t)$ where z is log-transformed size, and then wanting to plot the distribution $\tilde{n}(u, t)$ of untransformed size $u = \exp(z)$. Again, we change scales by computing the same thing two ways. $\tilde{n}(u, t)h$ is the number of individuals in the interval $[u, u + h]$, for small h. That corresponds to the size interval from $z = \log(u)$ to $z = \log(u + h)$. To find the width of that interval we use the Taylor series

$$\log(u + h) = \log(u) + h/u + \cdots .$$

The z-interval width is therefore h/u (for small h), so the number of individuals in the interval is $n(z, t)h/u = n(\log(u), t)h/u$. We conclude that

$$\tilde{n}(u, t)h = n(\log(u), t)h/u$$

and therefore $\tilde{n}(u, t) = n(\log(u), t)/u$.

The general recipe is this. Let g be the function taking the scale u on which you want to plot, to the scale z where the IPM operates. Then $\tilde{n}(u, t) = g'(u)n(g(u), t)$. In the case $z = \log(u)$, $g = \log$ and $g'(u) = 1/u$.

2.10 Appendix: Constructing IPMs when more than one census per time year is available

Here we outline a general approach for arriving at the "right" IPM in situations where more than one census (defined as any period of data collection) is performed each year. Using our Soay IPM as a case study, we show how this approach should be applied when one or more censuses are imperfect, in the sense that not every class of individual is measured. This is exactly the situation we face in the Soay system. Only lamb masses (measured shortly after birth) are acquired in the spring, whereas information about individuals of all ages are gathered in the late summer catch. As with models based on a single census per year, our aim is to show how to derive a model that correctly reflects the life cycle and census regime, and that can be parameterized from the available data.

The key to this methodology is to initially assume we have all the data we need at each census, and specify the corresponding model. We then "collapse" this model down in stages based upon our knowledge of which data are really available and our modeling objectives. In the Soay system, this means we first

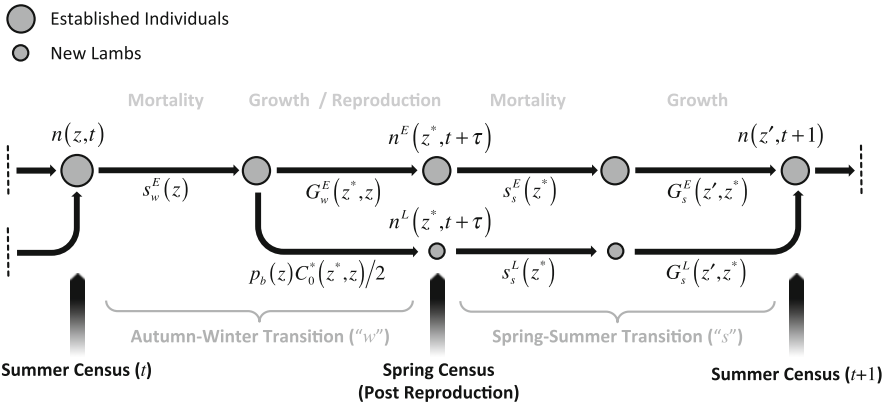

Fig. 2.12 Expanded life cycle diagram for the Soay sheep. Two census points are shown: the summer census of size z individuals and a post reproduction spring census of size z^* individuals. The diagram shows the fate of established individuals, $n^E(\ldots)$, and new lambs, $n^L(\ldots)$ over the autumn-winter transition (w subscript) and the spring-summer transition (s subscript). The life cycle is conceptualized as occurring in 4 phases: (1) an autumn-winter mortality phase; (2) an autumn-winter growth and reproduction phase; (3) a spring-summer mortality phase; (4) a spring-summer growth phase. These processes are described by survival functions $s_\circ^\circ(\ldots)$, growth kernels $G_\circ^\circ(\ldots)$, a recruit size-distribution kernel $C_0^*(z^*,z)$ and a probability of reproduction function $p_b(z)$.

need to construct a model that projects the dynamics from late summer to spring (post reproduction), and then from spring to late summer again. To do this, we will need to keep track of both established individuals (denoted with a superscript E) and new lambs (denoted with a superscript L). The established individuals' class includes every individual that survives to their first August catch. We distinguish winter and spring components of the demography with subscripts w and s, respectively. Spring size is denoted z^*. The remaining notation follows the conventions introduced in the main text unless otherwise stated (Figure 2.12).

As before, we aim to construct a model that projects the total August (summer) population size distribution, $n(z,t)$, between years. All individuals present at an August census are, by definition, established individuals. We begin with the equations for the spring size distributions of established individuals, $n^E(z^*,t+\tau)$, and new lambs, $n^L(z^*,t+\tau)$, produced by the August population at time t. These involve a winter survival-growth kernel, $P_w^E(z^*,z)$, and a winter fecundity kernel, $F_w^E(z^*,z)$, such that

$$\begin{aligned} n^E(z^*,t+\tau) &= \int_L^U P_w^E(z^*,z)n(z,t)dz, \\ &\text{where } P_w^E(z^*,z) = s_w^E(z)G_w^E(z^*,z) \\ n^L(z^*,t+\tau) &= \int_L^U F_w^E(z^*,z)n(z,t)dz \\ &\text{where } F_w^E(z^*,z) = s_w^E(z)p_b(z)C_0^*(z^*,z)/2 \end{aligned} \tag{2.10.1}$$

These expressions are very similar to the two components of the kernel from the Soay case study with one census per year, though everything in them now refers only to the winter transition. The fecundity component of the model does not contain a recruitment probability, $p_r(z)$, because this pertains to the next transition. We are also working with a different offspring size kernel, denoted C_0^* (not C_0), which describes the spring (not summer) mass of lambs.

The spring transition only involves survival and growth. The life cycle diagram tells us that the spring components of the model are

$$\begin{aligned} P_s^E(z', z^*) &= s_s^E(z^*) G_s^E(z', z^*) \\ n^E(z', t+1) &= \int_L^U P_s^E(z', z^*) n^E(z^*, t+\tau) dz^* \\ P_s^L(z', z^*) &= s_s^L(z^*) G_s^L(z', z^*) \\ n^L(z', t+1) &= \int_L^U P_s^L(z', z^*) n^L(z^*, t+\tau) dz^* \end{aligned} \tag{2.10.2}$$

The total August density function next year is then

$$n(z', t+1) = n^E(z', t+1) + n^L(z', t+1).$$

We now have a *consistent* model that properly accounts for the life cycle and known census times. This model projects from one summer to the next using two transitions: summer to spring and then spring to summer. It is therefore the IPM equivalent of a seasonal (or more generally, periodic) matrix projection model.

If we really had measured the size of all individuals in the spring, we could stop the model derivation here and begin to parameterize the various component functions from the data. However, in reality this is not a *feasible* model because only lambs were measured in the spring. There is no simple way to parameterize the established individuals' spring survival function, $s_s^E(z^*)$, and growth kernel, $G_s^E(z', z^*)$, in terms of spring size, z^*.

The solution to this problem is to collapse the survival-growth components of established individuals into a single transition. Instead of two survival-growth transitions governed by $P_w^E(z^*, z)$ and $P_s^E(z', z^*)$, we have to model the summer-summer transition in a single step, such that

$$\begin{aligned} n^E(z', t+1) &= \int_L^U P^E(z', z) n^E(z, t) dz \\ P^E(z', z) &= s_w^E(z) s_s^E(z) G^E(z', z) \end{aligned} \tag{2.10.3}$$

In this new formulation we have combined the two growth phases into one summer to summer growth kernel, $G^E(z', z)$, that can be parameterized from the summer catch data. We are still separating the winter and summer survival processes, though now they must be expressed as functions of the preceding

summer mass, z. Information about mortality is gathered over most of the year from regular visual censuses and by searching for dead individuals. Capture-mark-recapture methods could therefore potentially be used to estimate $s_w^E(z)$ and $s_s^E(z)$. However, this type of analysis is time-consuming and considerable expertise is needed to deal with missing individual size data (Langrock and King 2013).

What else can be done to simplify the model? Some more knowledge of the system helps here. We know that virtually all the mortality of established individuals is experienced during the winter transition. This means that it is reasonable (and convenient) to set $s_s^E(z) = 1$. After combining equation (2.10.3) and the lamb components of equations (2.10.1) and (2.10.2), following a little rearranging we get

$$\begin{aligned} n(z', t+1) &= \int_L^U [P(z', z) + F(z', z)] n(z, t) dz \\ P(z', z) &= s_w^E(z) G^E(z', z) \\ F(z', z) &= s_w^E(z) p_b(z) \int_L^U [s_s^L(z^*) G_s^L(z', z^*) C_0^*(z^*, z)/2] dz^* \end{aligned} \tag{2.10.4}$$

Now let $s_s^L(z^*) = p_r$ (i.e., a constant), $s_w^E(z) = s(z)$, and $G^E(z', z) = G(z', z)$; and define $C_0(z', z) = \int G_s^L(z', z^*) C_0^*(z^*, z) dz^*$. This is essentially the same model we described in our example. The only difference is that here we have incorporated lamb spring-summer growth into the mathematical details of the model, whereas in the original example we estimated $C_0(z', z)$ directly and allowed the data to effectively "do the integration" for us.

An important idea revealed by the derivation of the example model is that it will often be possible to construct a range of models of different complexity, particularly if more than one census is available. The choice of final model obviously depends upon the aims motivating its construction. If a model is being developed primarily to project the dynamics, e.g., to compare the population growth rates at different sites or explore transient dynamics, then it is reasonable to adopt the simplest possible model. This minimizes the number of parameters we have to estimate and keeps the implementation simpler. If, on the other hand, the model is to be used to understand selection on the life history, e.g., via the calculation of parameter sensitivities, then it makes more sense to keep the components of lamb demography separated as in equations 2.10.4. This provides more insight into the effect of reproduction on population growth rate. The formulation we outlined assumes that there are no maternal effects on lambs that play out over spring and summer. This assumption could be relaxed, though it results in a much more complicated model because we have to keep track of the bivariate mother-offspring size distribution. However, this effort might be warranted if the model is going to be used to understand how phenotypic maternal effects impact the dynamics.

If we really had gathered size data for individuals of all ages at both censuses it would be natural to construct a seasonal IPM. Alternatively, we could "collapse" the model to project the dynamics among years. We would then have to make a decision about which census point to use: spring to spring or summer to summer. The kernels would be different, but the resultant dynamics would be essentially the same. In both cases the model would be of the post-reproductive census variety, because both reproduction kernels contain survival functions of the established individuals. What if we had collected spring size data prior to, instead of after, reproduction? In this case we could construct either a post-reproductive census model (by projecting from summer to summer) or a pre-reproductive census model (by projecting from spring to spring). This illustrates another feature of IPMs constructed from multiple censuses. The pre- versus post-reproductive census distinction is no longer necessarily a feature of the data collection methodology. Instead it reflects the life cycle, the data, and our modeling decisions.

http://www.springer.com/978-3-319-28891-8

Data-driven Modelling of Structured Populations
A Practical Guide to the Integral Projection Model

Ellner, S.P.; Childs, D.Z.; Rees, M.

2016, XIII, 329 p. 67 illus., 29 illus. in color., Softcover
ISBN: 978-3-319-28891-8