Integrated population models

tentative title

Michael Schaub & Marc Kéry

Draft of a book
to be published by *Academic Press*

1 June 2018

vogelwarte.ch

Table of contents

3. Introduction to stage-structured population models

3.1. Introduction

A central goal in population ecology is to achieve a quantitative understanding of a study population. We may look into the past to understand why a population has experienced a particular dynamics (a retrospective analysis) or we may look into the future to predict its likely future dynamics (a prospective analysis). Some population studies focus exclusively on the *number* of individuals (population size) and its change over time. The primary interest is the population growth rate, which is the main parameter of the corresponding models. Population trends which are the longer-term sustained rate of population changes can be calculated as well (REF). Extensions of such models include density-dependence, environmental and demographic stochasticty (Lande et al. 2003, Dennis and Taper 1994, Dennis et al. 1991) and observation errors (Dennis et al. 2006, Pasinelli et al. 2011). Other studies aim to obtain a more mechanistic understanding of population dynamics, which requires the inclusion of demographic rates. Such models are extensions of these more phenomenological models just described. Typical research aims of these more mechanistic models are the assessments of how strongly each demographic rate contributes to the population growth and how the latter is expected to change in the future if the value of a demographic rate or its temporal variability changes. Clearly such population models must include an explicit link between demographic rates and population structure and size – the population growth rate appears as a derived quantity. A number of excellent text books about demography and population dynamics have appeared that deal with this kind of population model (Caswell 2001, Conroy and Carroll 2009, Newton 1998, Morris and Doak 2002, Mills 2013, Skalski et al. 2005), the most detailed one being the treatise by Caswell (2001). Several names exist for these models, including *matrix population model*, *matrix projection model, matrix-based projection models, stage-structured population model, matrix model* or *Leslie matrix model*. In this book we will use the term matrix population models, because this seems to be a general and simple term. Integrated population models also make an explicit link between demography and population size and structure, hence matrix population models are at the core of any IPM and a solid knowledge of them is compulsory when you want to apply IPM techniques. Therefore, in this chapter we provide a concise introduction to matrix population models. Of course, we make no claims to be fully comprehensive about this huge topic and for (much) more information, you should get a copy of Caswell (2001).

A population of size N_t in year t can typically be thought to be composed of different types of individuals regarding their origin in year t-1. They may be survivors from last year or they may be new in the population, and if so, they can either be local recruits or immigrants. Thus, the population size can be written as the sum of these three components

$$N_t = R_t + I_t + S_t$$

Here, R_t is the number of local recruits, i.e., individuals born in the same population in year t-1 and that survived and did not emigrate from the population until year t. I_t is the number of immigrants, i.e., individuals born outside of the study population in year t-1 or earlier that have so far never attempted to immigrate and reproduce in the study area. Finally, S_t is the number of survivors from year t-1, i.e., individuals that were already present in the population in year t-1 and survived until year t and did not emigrate to another population. It is possible to further elaborate on this and express the number of survivors as the difference between the population size in year t-1, the number of individuals that have died (D_t) and the number of emigrants (E_t):

$$S_t = N_{t-1} - D_t - E_t \, .$$

The combination of these two equations leads to:

$$N_t = N_{t-1} + R_t + I_t - D_t - E_t$$

Hence, in words, the population size in year *t* is given by the population size the year before, plus the number of new individuals that are due to local recruitment (which were born on site) and to immigration, minus the number of individuals that are lost due to death and emigration.

The population growth rate (λ_t) is a key parameter to describe population dynamics. It is defined as the rate of change of the size of a population between years *t*-1 and *t*:

$$\lambda_t = \frac{N_t}{N_{t-1}} = 1 + \frac{R_t}{N_{t-1}} + \frac{I_t}{N_{t-1}} - \frac{D_t}{N_{t-1}} - \frac{E_t}{N_{t-1}}$$

The ratios on the right hand side of the equation are *per capita* numbers and represent the demographic rates. Written as such, we get

$$\lambda_t = \frac{N_t}{N_{t-1}} = 1 + r_t + i_t - d_t - e_t$$

Hence, the population growth rate is 1 plus the recruitment rate (r_t) and the immigration rate (i_t), minus the mortality rate (d_t) and the emigration rate (e_t). The survival rate (s_t) is 1 minus the mortality rate, hence, we can also write

$$\lambda_t = \frac{N_t}{N_{t-1}} = r_t + i_t + s_t - e_t .$$

We make a number of important observation here. First, it is obvious that the rate of population change, the population growth rate, is a function of exactly four demographic rates: recruitment, immigration, survival and emigration. Hence, whenever a population growth rate changes, this must be due to the change of at least one of the four demographic rates. Thus, this equation provides a formal link between demography and population growth, which allows us to study demographic causes of population change or population stability. However, not every change in a demographic rate must necessarily result in a change of the population growth rate. For instance, if one rate increases and another decreases by the same amount, then we observe no change in the population growth rate, although the demography of the population has changed. Thus, changes in the demographic rates are a necessary, but not sufficient condition for a change of the population growth rate. Second, the population can change in *size* even if the population growth *rate* does not. To see this, consider a population with population growth rate λ that is constant over time and calculate population size in each year. Starting with population size of N_0 the population size in year 1 is $N_1 = \lambda N_0$, in year 2 it is $N_2 = \lambda N_1 = \lambda^2 N_0$ and in year *t* $N_t = \lambda N_{t-1} = \lambda^t N_0$. Only if $\lambda = 1$ does the population size not change, but if $\lambda > 1$ the population grows exponentially and if $\lambda < 1$ it declines asymptotically to zero (Fig. 3.1.). Finally, we note a "notational conflict" around the population growth rate, which is traditionally expressed as λ, as is the parameter of a Poisson distribution. Later in the book, we will see models with both a growth rate and a Poisson expectation and we will use λ for the former only.

Up to now we have assumed that demographic rates were constants and did not vary over time or other dimensions. Temporal variation of the demographic rates is particularly common. For example, annual survival of barn owls is much lower in years with harsh winters than in years with mild winter (Altwegg et al. 2003) or reproduction of fat dormice only occurs in years with beech mast seeding (Pilastro et al. 2003), resulting in temporal variation of annual survival or reproductive output, respectively. Although the resulting population size is fluctuating, it is still growing exponentially. However, real-world populations cannot grow for ever, but instead there must be a feedback between their size and the demographic rates in order to keep them within bounds and avoid population explosion or extinction. This feedback represents the regulation of a population. A fundamental aim in population ecology is to understand how populations are regulated, i.e. to identify the limiting factors of population growth (Newton 1998). One fundamental mechanisms providing this feedback is *density-*

dependence. For our population model this means that at least for one of the demographic rates there must be a negative relationship with population size. Several mechanisms may be responsible for such a negative feedback. An important example is competition where individuals of territorial species are forced to occupy territories of lower and lower quality as a population grows, thereby decreasing the average territory quality which results in a decline of productivity at the population level (Rodenhouse et al. 1997). Competition may also result in smaller territory size at high population density as opposed to low density and as a consequence the available amount of resources is reduced for the average individual which may again lead to reduced productivity (Sillett et al. 2004, Rodenhouse et al. 2003). Of course, not only productivity, but also survival may change as a function of density, e.g. if wintering individuals are forced to stay at lower quality sites due to compentition (Gill et al. 2001, Newton 2004). In these cases we speak of density-dependent productivity or survival, but indeed there may a density-dependent feedback loop of population density with any of the demographic rates above. A positive relationship between a demographic rate and population size is also possible (called "Allee effect"; Courchamp et al. 1999), but is typically restricted to small population sizes (Kramer et al. 2009).

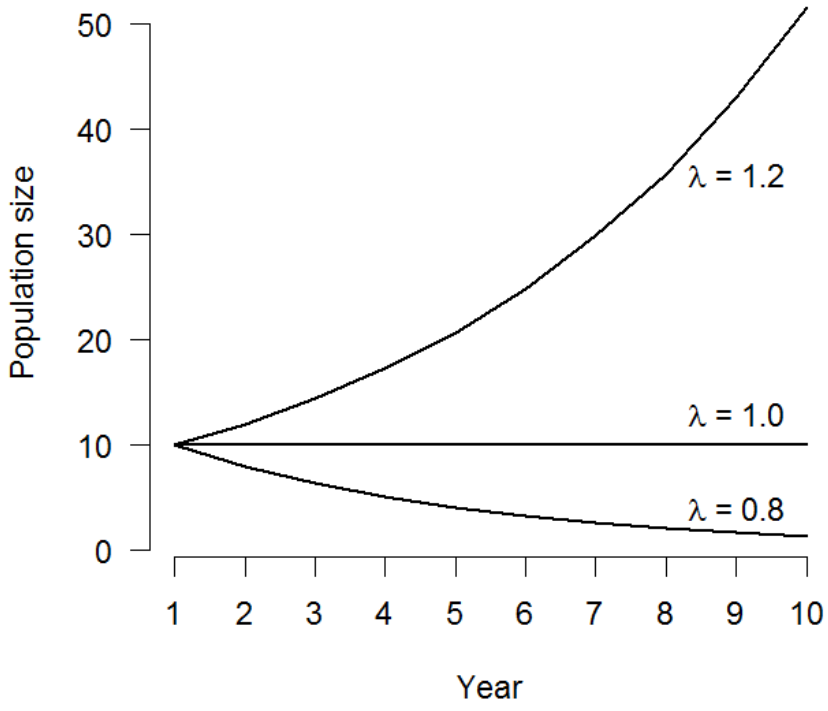


Figure 3.1 Development of exponentially growing populations with three different values of the population growth rate (λ) which are constant over time. Only when the population growth rate is exactly equal to 1 does the population size not change, otherwise the population either increases exponentially ("explodes") or declines asymptotically to zero (goes extinct).

3.2. Age- and stage-structured population models

The model in eq. 1 is one of the simplest population models possible and only differentiates between different types of individuals in terms of "where they come from". This model may be useful in some circumstances, but it makes the very restrictive assumption that all individuals in the population are identical from a demographic point of view. Thus, all individuals in the population must have identical survival probabilities, produce the exact same number of offspring and have identical emigration probabilities. Clearly, these assumptions of identical demographic rates for all individuals can never be strictly true. For example, survival typically changes with the age of an individual – it increases from birth, reaches a maximum and declines thereafter (i.e., there is senescence; Gaillard et al. 1994, Loison et al. 1999, Tavecchia et al. 2001). Reproductive rates often also change with age. Many species reach sexual maturity only after two or more years (Harvey and Zammuto 1985, Hadley et al. 2006) and after the start of reproduction, reproductive performance may increase with age and then decline again at older ages (Forslund and Pärt 1995, Aubry et al. 2009). Yet, age is not the only factor that usually has a large impact on demographic rates. For example many sexually mature albatrosses are able to reproduce only every second year (Converse et al. 2009, Gauthier et al. 2012), and thus the reproductive output of an individuals in year *t* depends on whether it had reproduced in year *t*-1, or put it in other words, in which state (breeder, skipped breeder) the bird was in year *t*-1. Another example is fish, growth…. In these cases the demographic performance is not only a function of age, but generally a function of a "stage". In the albatross example the stages are "breeder" and "non breeder" while in the XX example the stages are """". In most species, demography changes with age and/or stage and realistic population models ought to take account of this structure.

The model in eq. 1 also requires that time is discrete, that is, we look at the population at particular and predefined time points. The interval between these time points is called the projection interval. It is often one year in vertebrate species, but there is flexibility in defining other intervals. Population models that do not need discrete time points, but model the population size continuously over time are mathematically more demanding and less adequate for vertebrate species (Hastings 1997). Therefore, we focus in this book only on discrete-time models and do not touch continuous-time models.

Age-structured population models have a relatively long history and go back to Leslie (Leslie 1945) who formalised their mathematics. Stage-structured population models are a generalisation of age-structured population models and received their first mathematical treatment by Lefkovitch (1965); see Caswell (2001) for more on their history. Strictly speaking, a Leslie matrix model is an age-structured population model, while a stage-structured model is a Levkovitch matrix model, though we note that Leslie matrix model is often used exchangeably.

In the following we illustrate the construction of both an age- and a stage-structured population model.

Figure 3.2 Woodchat shrike male.

3.2.1. From a life cycle graph to population equations

A life cycle graph is a graphical depiction of the life cycle of a species and can be constructed for either age- and or stage-structured models. It is composed of nodes that stand for the age/stage-specific numbers of individuals in the population and of arrows that connect these nodes and which represent demographic transition rates. All nodes between which demographic transitions are possible within the defined projection interval are connected with an arrow. The arrows are labeled with the transition probabilities, which are given by demographic processes such as survival, reproduction or growth, or a combination thereof. In our experience it is very helpful to start the development of any age/stage-strucured population model, and thus of any IPM, with the construction of a life cycle graph, because it enforces more transparency with respect to the assumptions and simplifications that we make about the demography of our study species. Furthermore, the formulae to describe age/stage-specific population change can be derived very easily from such a graph. If the graph is correct, it is hard to make errors in the formulation of these equations.

We illustrate the creation of life cycle graphs and the derivation of the age/stage-specific population sizes using the beautiful woodchat shrike (*Lanius senator*, Fig. 3.2), a species that will feature prominently at various places in this book. Woodchat shrikes reproduce mainly in the Mediterranean region in Western Europe and during the winter migrate to sub-Saharan Africa (Lefranc and Worfalk 1997). This splendid bird mainly feeds on invertebrates (beetles, grasshoppers) and unfortunately its populations are declining in most of its distribution area (Sanderson et al. 2006). The woodchat shrike has a life cycle that is typical for many passerine bird species. It starts to reproduce when one year old. Fecundity (expressed as the annual number of fledglings per female of a given age class) is lower for 1-year old compared to older females. The annual survival probability changes with age - juvenile survival (from fledging to age one year) is lower than adult survival (from age 1 onwards). Before we can start to draw a life cycle graph based on this life history information of the shrikes, we have to take a number of decisions.

First, remember that in the models in this book we discretize time and hence we need to decide at which point of time we want to observe the population and how long is the projection interval. In birth-pulse populations, where reproduction occurs in one particular period of the year (and not continuously across the complete year as in birth-flow populations) the projection interval is typically one year and there are two basic options for the observations: we can observe the population either just before the reproductive period in which case we speak of a pre-birth-pulse model, often also called a model for a pre-birth pulse census or pre-breeding census. Alternatively, we can observe the population just after reproduction is completed and obtain a post-birth-pulse model, also referred to as a model for a post-birth-pulse census or post-breeding census. As we will see later, the inferences about the population dynamics are not affected by this decision. What is different though in these two types of models is the identity of the specific age/stage classes – some of them only occur in one type of model and not in the other. Therefore the decision about which type to choose depends on our research interest. For IPMs pre-birth-pulse models are often more practical because of an easier indexing of time and age (see below). In any case, it is important to stick to one of the two possibilities. In our experience students with limited previous exposure to these models often have difficulties not confuse them. Therefore, we spend some time on explaining them.

Depending on our research interest, we can also discretize time in a different manner. For example, we might be interested in modeling population size changes within a year and construct a seasonal model. Since the woodchat shrike is migratory, we may want to study the dynamics of population size not only at the breeding, but also at the wintering grounds (Hostetler et al. 2015). Such models of course require more detailed information about demographic rates, such as seasonal survival probabilities. We illustrate one example of such a model in chapter XY, but most of them are annual.

Second, we have to decide whether we include females only, or whether we want to add the males also and construct a two-sex model. Often, population models are female based, thus they include the females only and assume that the availability of males is unlimited for females, or expressed in a different way, that the males have no impact on the dynamics of the population. Of course this assumption is not strictly true, but for monogamous species (and sometimes for others with a more complex mating system as well) and when sexes have identical (similar) demographic rates it is often a useful approximation. In this book we will mostly use female based models, but have examples for two-sexes models in chapter XY. Two sex models require the formulation of a mating function and become more complex (Caswell 2001, chapter 17).

Third, we have to decide whether we will include dispersal and if so, if we have multiple populations between which we want to model exchanges. Models including such explicit descriptions of dispersal can easily become fairly complicated. Most models in this book do not explicitly include dispersal (but see chapter XX) and thus they only consider recruitment and survival as demographic processes. We therefore assume that individuals are completely philopatric, i.e., remain within a specified study area. Since dispersal is an effect of spatial scale, we could also say that we model the world population of our target species - then dispersal from and to this world population is of course impossible.

Equipped with this knowledge which includes fully understanding of these assumptions we are now ready to draw our first life cycle graph. For illustrative purposes we here develop life cycle graphs for woodchat shrikes for all possible types: i.e. age-structured pre-birth-pulse, stage-structured pre-birth-pulse, age-structured post-birth-pulse and stage-structured post-birth-pulse models, each with a projection interval of one year. We develop formulae to describe age/stage-specific population dynamics for all of them. In all cases we consider a

female-based model with a projection interval of one year and we have the following demographic rates:

- juvenile survival from t to t+1 ($s_{j,t}$),
- adult survival from t to t+1 ($s_{a,t}$)
- fecundity of 1-year old females in year t ($f_{1,t}$)
- and fecundity of older ("adult") females in year t ($f_{a,t}$)

Since the model is female-based, fecundity is defined as the production of female fledglings per reproducing female of a given age class.

3.2.2. Age-structured, pre-birth-pulse model

We start with the construction of the life cycle for an age-structured, pre-birth-pulse model. To make sure that we do not confuse the pre- and post-birth pulse models, we find it useful to always start with the definition of the youngest age class. In the case of a pre-birth-pulse model we look at our population *just before* reproduction starts, and thus the youngest individuals are approximately 1 year old (not exactly, but very close to). We label the node of this age class *1y*. Clearly we need more nodes in our graph, but how many? For a strictly age-strucured model we have to define the oldest possible age class, i.e., we must assume that no individual can gets older than this age class. This may seem like a difficult decision, but in practice we simply can add more than enough age classes; having too many does not harm. For the population model that we develop here let's assume that woodchat shrikes can reach a maximal age of 4 years. Admittedly this may be a bit low, but we make this choice partly for illustrative purposes to avoid the graphs and formulae to become too unwieldy. It would be easy to extend the maximal age to, say, 10 years, which in reality for this species may be more adequate. Once we have defined the youngest and the oldest age classes, we know that our graph has four nodes and we label them *1y*, *2y*, *3y*, and *4y* (which make obvious their meaning). We here show two variants of life cycle graphs. In the first variant (Fig. 3.3.) which we call the *full life-cycle graph*, we draw nodes for each year. The labels of these nodes remain the same in all years, but the number they represent may changes from one year to the next.

Next we connect the nodes with arrows. Let's start with the survival part, since this is easier. From year t to $t+1$, one-year old individuals can only make the transition to the 2-year old individuals, no other transitions is possible (e.g. they cannot remain in the 1-year age class, nor can they transition to become a 3-year old in one time step). The probability associated with this transition is the adult survival probability ($s_{a,t}$), and we label the arrow accordingly. Next, we draw the other survival transitions from nodes *2y* and *3y* in year t to nodes *3y* and *4y* in year t+1, respectively, and label them $s_{a,t}$. Since we have decided that our shrikes can at most become 4 years old, there is no arrow leaving node *4y*.

After drawing the survival arrows, we go on to drawing the arrows representing recruitment. This can be a little trickier, because recruitment is composed of two underlying demographic processes: fecundity and survival. Since *1y* individuals already reproduce, there is an arrow from node *1y* in year t to node *1y* in year t+1, but how do we label it? We find it useful to start with the demographic processes in chronological order. In a pre-birth-pulse model, the first process is reproduction, thus we have $f_{1,t}$. Be careful with the index for time and age: in this graph reproduction happens in year t and it refers to 1-year individuals. The next process is the survival of the fledglings produced, which is $s_{j,t}$. The label of the arrow is the product of the two, thus $f_{1,t}\, s_{j,t}$. Individuals that are 2-years old also contribute to recruitment, hence there is an arrow from node *2y* in year t back to node *1y* in year t+1. Productivity of 2-year old individuals in year t is $f_{a,t}$. Fledglings must survive until year t+1 to enter the class of the *1y* birds, and thus the complete transition is labelled $f_{a,t}\, s_{j,t}$. The same recruitment applies also for individuals that

are over 2-years old, since they all have the same productivity. Now we have described all possible transitions from year *t* to year *t*+1. The transitions from year *t*+1 to year *t*+2 are analogous, only the year index of the demographic rates is different (*t*+1 instead of *t*, Fig. 3.3.a).

Usually we draw a simplified form of the life cycle graph (Fig. 3.3.b). The key difference to the full life cycle graph is that redundant nodes in different years are collapsed. Thus, a transition from node *A* to node *B* always means the transition from node *A* in year *t* to node *B* in year *t*+1 in the simpler form the of the life cycle graph. Because of this, self loops (say, from *A* to *A*) can occur in the simplified life cycle graph. In our example a self-loop is present for node *1y*, because 1-year old individuals in year *t* can produce 1-year old individuals in year *t*+1.

The development of the formulae for the age-specific population sizes in year *t*+1 given the population sizes in year *t* and the demographic rates is now trivially easy: to get the number of individuals of a particular node, we have to sum over all the arrows that enter this node and each term in that sum is the product of the node from where the arrow originates and the transition probability, which itself will consist of one or the product of multiple demographic rates. This applies regardless of whether we use the full life-cycle or the more common simplified life cycle graph. Let's denote the age-specific population sizes with *N*, thus the number of 1-year old individuals (hence node 1y) is year *t* is $N_{1y,t}$. The age-specific population sizes in year *t*+1 are then:

$$N_{1y,t+1} = N_{1y,t}f_{1,t}s_{j,t} + N_{2y,t}f_{a,t}s_{j,t} + N_{3y,t}f_{a,t}s_{j,t} + N_{4y,t}f_{a,t}s_{j,t}$$
$$N_{2y,t+1} = N_{1y,t}s_{a,t}$$
$$N_{3y,t+1} = N_{2y,t}s_{a,t}$$
$$N_{4y,t+1} = N_{3y,t}s_{a,t}$$

Simple and very logical, isn't it? We can write exactly the same model using linear algebra in vector and matrix form. The population size is now a vector whose elements are the age-specific population sizes. We have to find a matrix in such a way that the multiplication of this matrix with the population size vector of year *t* yields the population size vector in year *t*+1 (as above). We then get

$$\begin{bmatrix} N_{1y,t+1} \\ N_{2y,t+1} \\ N_{3y,t+1} \\ N_{4y,t+1} \end{bmatrix} = \begin{bmatrix} f_{1,t}s_{j,t} & f_{a,t}s_{j,t} & f_{a,t}s_{j,t} & f_{a,t}s_{j,t} \\ s_{a,t} & 0 & 0 & 0 \\ 0 & s_{a,t} & 0 & 0 \\ 0 & 0 & s_{a,t} & 0 \end{bmatrix} \begin{bmatrix} N_{1y,t} \\ N_{2y,t} \\ N_{3y,t} \\ N_{4y,t} \end{bmatrix}.$$

The transition matrix is the famous Leslie matrix, it is composed of demographic rates. The population model can also be written as $\boldsymbol{N}_{t+1} = \boldsymbol{A}_t \times \boldsymbol{N}_t$ where $\boldsymbol{N}_t$ is the population vector in year *t*, $\boldsymbol{A}_t$ is the Leslie matrix and x denotes the matrix multiplication operation. Note that unlike in ordinary multiplication for two scalars, the order of the elements in a matrix multiplication *does* matter; see later. All these different ways describe the same process, thus they are fully equivalent. However, in integrated population models we will need the first formulation where we have an equation for each age class that we defined in the population.

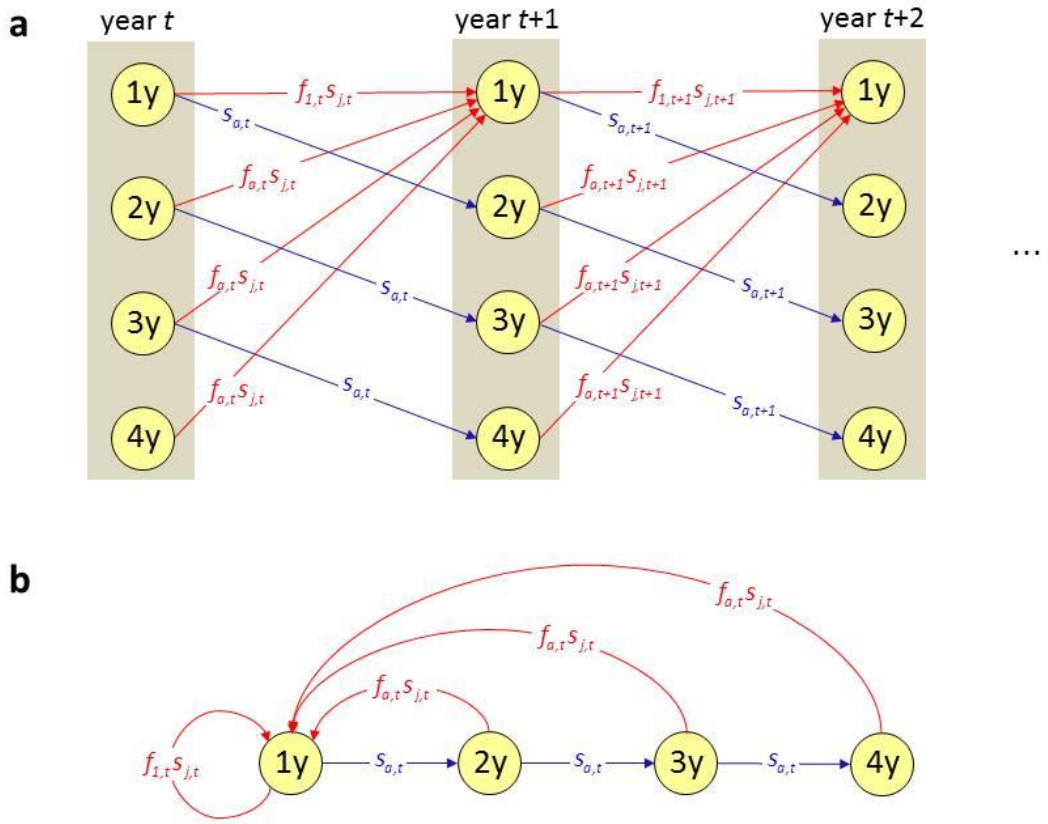


Figure 3.3 Two variants of a life cycle graph of an age-structured, pre-birth-pulse model for the woodchat shrike; a) the full life cycle graph, b) the simplified life-cycle graph. The maximal age of a shrike is assumed to be 4 years. Recruitment processes, containing a contribution of fecundity, are depicted in red, while pure survival processes are shown in blue.

3.2.3. Stage-structured, pre-birth-pulse model

The age-structured population model has two disadvantages: that the maximal age must be defined and that it requires the explicit inclusion of all age classes until the maximal age. Depending on the species, this may result in very large life cycle graphs, a large number of equations or a high-dimensional Leslie matrix. In the life cycle graphs (Fig. 3.3.) we see that individuals in nodes *2y* and *3y* are assumed to be demographically identical; all the arrows leaving and entering the node are identical. Therefore we can lump these nodes. We label this new node *ad*, since it refers to "adult" birds in the sense that they are older than 1 year. If we also relax the assumption that the maximal age is 4 years and instead assume that survival beyond age 4 is $s_{a,t}$ and reproduction remains at $f_{a,t}$, then we can also subsume node *4y* into the new node *ad*. A transition from node *ad* in year *t* to node *ad* in year *t*+1 becomes now possible, meaning that there is no longer any predefined maximal age for the shrikes. The resulting full and simplified life cycle graphs are shown in Fig. 3.4. This is now a *stage-structured* life-cycle graph, because we do no longer classify the individuals by chronological age only, but rather by the two stages *1y* and *ad*. Strictly speaking the model contains a mixture of a pure age class (*1y*) and of a stage that contains multiple age classes (*ad*). In the simple life cycle graph there is now an additional self-loop on the node *ad*.

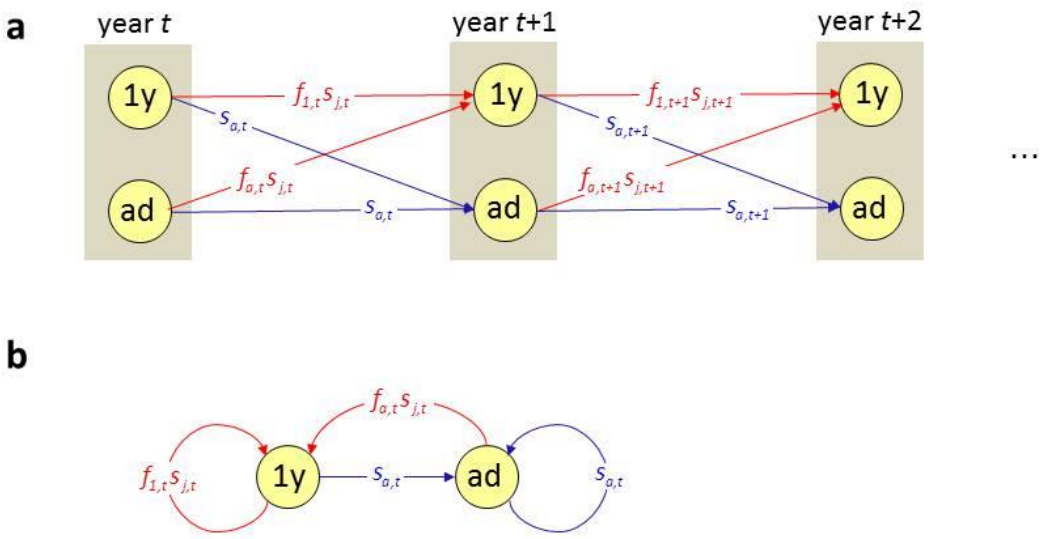


Figure 3.4 Life cycle graphs of a stage-structured, pre-birth-pulse model for the woodchat shrike example; a) shows the full life cycle graph, b) the simple life-cycle graph. Recruitment processes are depicted in red, pure survival processes in blue.

Exactly in the same way as for the age-structured model before, we can easily derive the formulae for the stage-specific population sizes:

$$N_{1y,t+1} = N_{1y,t}f_{1,t}s_{j,t} + N_{ad,t}f_{a,t}s_{j,t}$$
$$N_{ad,t+1} = N_{1y,t}s_{a,t} + N_{ad,t}s_{a,t}$$

Also the linear algebra version of the model is fully analogous to that before:

$$\begin{bmatrix} N_{1y,t+1} \\ N_{ad,t+1} \end{bmatrix} = \begin{bmatrix} f_{1,t}s_{j,t} & f_{a,t}s_{j,t} \\ s_{a,t} & s_{a,t} \end{bmatrix} \begin{bmatrix} N_{1y,t} \\ N_{ad,t} \end{bmatrix}$$

Not surprisingly the stage-structured population model looks simpler than the fully age-structured model. The transition matrix of stage-structured models is often referred to as a Lefkovitch matrix (Lefkovitch 1965). Note that it is always possible to turn an age-structured population model into a stage-structured one.

3.2.4. Age-structured, post-birth-pulse model

Next we move to a post-birth-pulse model (Fig. 3.5.a), where we observe (or "census") the population immediately after reproduction and thus the youngest individuals are juveniles (fledglings), which we denote as *juv*. In the age-structured model we again make the assumption that the maximal possible age of the shrikes is 4 years, i.e. the last possible reproduction occurs at age 4 years. In the post-birth-pulse model the oldest age class is therefore node *3y*. This may seem confusing, isn't it? The reasons why the oldest age class is *3y* is because individuals have to survive first one year, they become 4 year old and then reproduce for the last time. Thus, although we have only node *3y* in our model, the individuals are 4 years old when the reproduce for the last time, which is fully consistent with our assumption about the maximal age. The four nodes that we define are *juv*, *1y*, *2y* and *3y*. We draw the arrows and again start with those representing survival. From node *juv* to *1y* there is an arrow labelled $s_{j,t}$, because it refers to the survival of juvenile individuals. From nodes *1y* and *2y* in year *t* to nodes *2y* and *3y* in year *t*+1 we have adult survival $s_{a,t}$. For the recruitment we have an arrow (transition) from node *juv* in year *t* to node *juv* in year *t*+1. To get the label of the arrow correctly, start again chronologically with the demographic processes involved. We first have a survival process, thus the *juv* individual in year *t* must survive until the next breeding season and does so with probability $s_{j,t}$. Next, we have reproduction. The surviving individual is now 1-year old and reproduction occurs in year *t*+1. Therefore, the correct reproduction parameter is $f_{1,t+1}$, and the complete label of the arrow is $s_{j,t}f_{1,t+1}$ (note the different time index for *s* and *f* !). The other recruitment arrows are similar, the only difference is that survival refers to individuals that are at least one year old and therefore $s_{a,t}$ applies and that fecundity refers to females that are older than one year. You can see that the construction of a post-birth-pulse model is more tricky, because the indexing of age and time is more involved. On the other side the post-breeding model may be more natural because it includes the juveniles as a proper age class, while in the pre-breeding model juveniles do not explicitly appear.

Based on the full life cycle graph the construction of the simple life cycle graph is not a big challenge (Fig. 3.5.b). The post-birth-pulse and pre-birth-pulse life cycle graphs look the same regarding the number of nodes and type of arrows in our example, the only differences are the labels of the nodes and arrows. This need not hold in any case, and there are examples where the number of nodes differs between post-birth and pre-birth-pulse models for the same population.

The derivation of the age-structured population sizes is easy from the life cycle graph:

$$N_{juv,t+1} = N_{juv,t}s_{j,t}f_{1,t+1} + N_{1y,t}s_{a,t}f_{a,t+1} + N_{2y,t}s_{a,t}f_{a,t+1} + N_{3y,t}s_{a,t}f_{a,t+1}$$
$$N_{1y,t+1} = N_{juv,t}s_{j,t}$$
$$N_{2y,t+1} = N_{1y,t}s_{a,t}$$
$$N_{3y,t+1} = N_{2y,t}s_{a,t}$$

Written in linear algebra with population vectors and the Leslie matrix we obtain:

$$\begin{bmatrix} N_{juv,t+1} \\ N_{1y,t+1} \\ N_{2y,t+1} \\ N_{3y,t+1} \end{bmatrix} = \begin{bmatrix} s_{j,t}f_{1,t+1} & s_{a,t}f_{a,t+1} & s_{a,t}f_{a,t+1} & s_{a,t}f_{a,t+1} \\ s_{j,t} & 0 & 0 & 0 \\ 0 & s_{a,t} & 0 & 0 \\ 0 & 0 & s_{a,t} & 0 \end{bmatrix} \begin{bmatrix} N_{juv,t} \\ N_{1y,t} \\ N_{2y,t} \\ N_{3y,t} \end{bmatrix}.$$

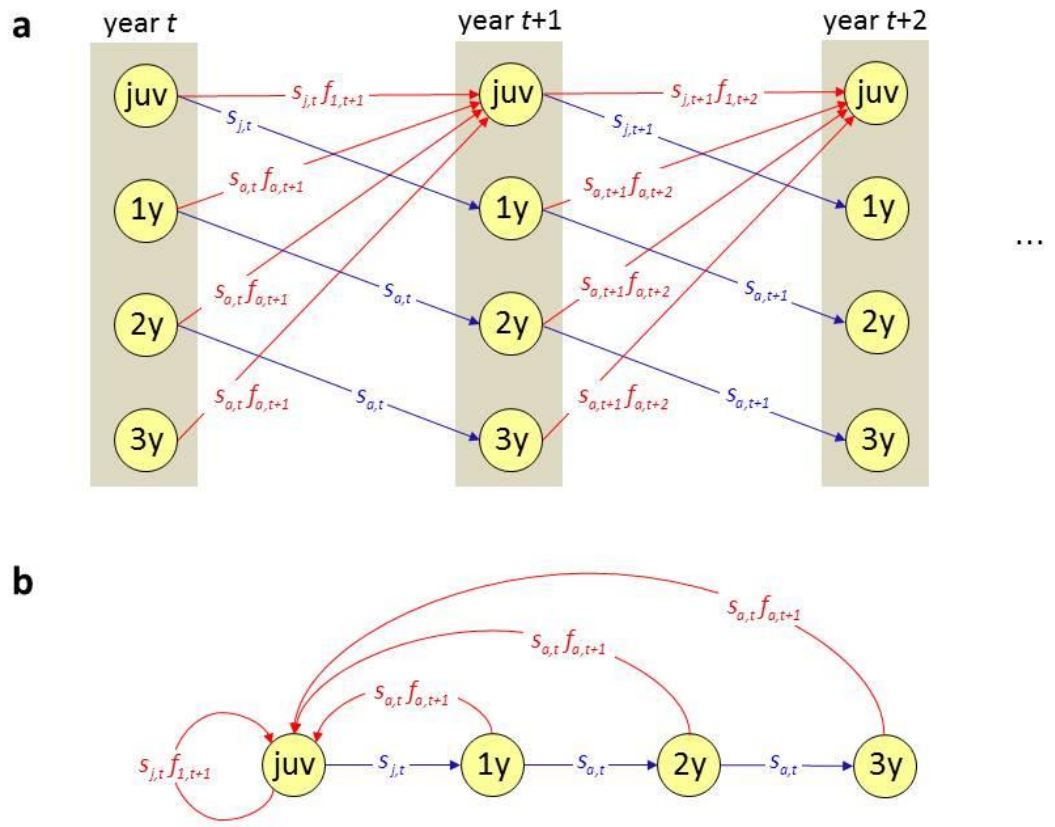


Figure 3.5 Life cycle graphs of an age-structured, post-birth-pulse model for the woodchat shrike example; a) shows the full life cycle graph, b) shows the simple life-cycle graph. The maximum age is assumed to be 4 years. Recruitment processes are depicted in red, pure survival processes in blue.

3.2.5. Stage-structured, post-birth-pulse model

Finally, we develop a stage-structured, post-birth-pulse model. In the full life-cycle graph we can first add an arrow from node *3y* in year *t* to node *3y* in year *t*+1 and in the simple life-cycle graph a self-loop to node *3y*. This relaxes the assumption of the maximum age of the shrikes. We then recognize that all arrows leaving and entering nodes *1y*, *2y* and *3y* are identical and can be lumped. We label the resulting new node *ad*. The corresponding full and simple life-cycle graphs are shown in Fig. 3.6.

The derivation of the formulae for the stage-specific population sizes poses no problem and we get

$$N_{juv,t+1} = N_{juv,t}s_{j,t}f_{1,t+1} + N_{ad,t}s_{a,t}f_{a,t+1}$$

$N_{ad,t+1} = N_{juv,t}s_{j,t} + N_{ad,t}s_{a,t}$

or, using linear algebra:

$$\begin{bmatrix} N_{juv,t+1} \\ N_{ad,t+1} \end{bmatrix} = \begin{bmatrix} s_{j,t}f_{1,t+1} & s_{a,t}f_{a,t+1} \\ s_{j,t} & s_{a,t} \end{bmatrix} \begin{bmatrix} N_{juv,t} \\ N_{ad,t} \end{bmatrix}.$$

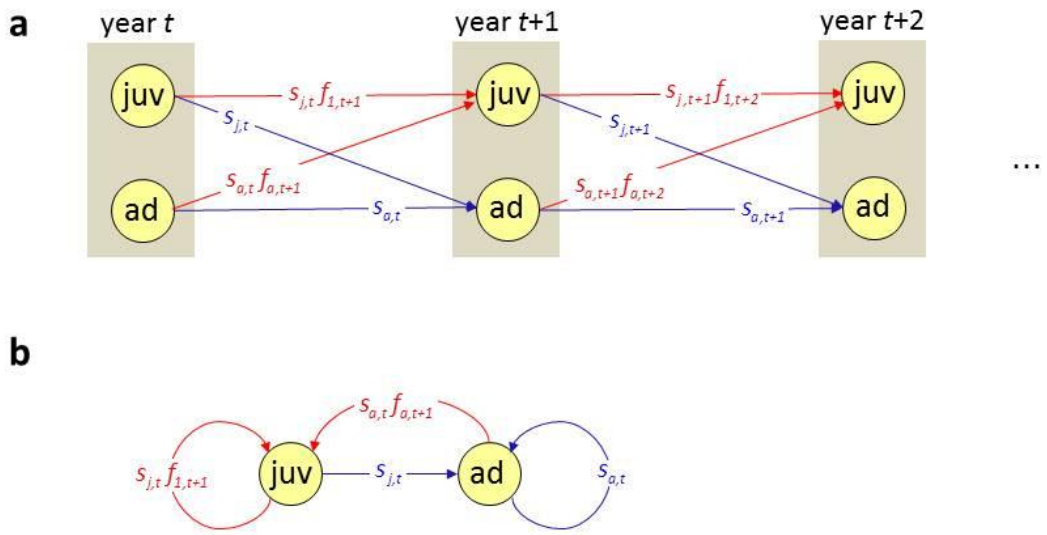


Figure 3.6 Life cycle graphs of a stage-structured post-birth-pulse model for the woodchat shrike example, a) shows the full life cycle graph, b) the simple life-cycle graph. Recruitment processes are depicted in red, pure survival processes in blue.

All the equations that we have derived so far allow the calculation of the number of birds in each age/stage-specific class of the shrike population at any given discrete point of time, which are the time when the birds start egg laying in each year (pre-breeding census) or when the birds fledge in each year (post-breeding census). We need to know the demographic rates and the age/stage-structured population sizes referring to these discrete points of time. The total population size at time *t* is then simply the sum of the age/stage-specific population sizes at time *t*. Provided that the demographic rates do not change as a function of population size (i.e. there is no density-dependence), the total population sizes as well as the age/stage-specific population sizes are then all growing asymptotically at the same rate (the population growth rate) – the population either shows exponential increase or decline as already shown in Fig. 3.1. With age/stage-structured population models, or more generally with matrix population models, we have defined an explicit link between population size and demography, and such models are the base of nearly all vertebrate population modeling studies with a birth-pulse. In

contrast, population models of species with a birth-flow (i.e. reproduction may occur more or less continuously during a calendar year) require continuous time model that are more complex and not treated here. As an approximation that does not require a continuous time model birth-pulse models with a smaller projection interval than one year can be used. Birth-flow population are generally much less typical for most temperate vertebrates.

3.3. Classical analysis of a matrix population model

Common aims in the analysis of age- and stage-structured population models, or more generally of matrix population models, are the calculations of quantities that can be used to describe the population dynamics (life history summaries), to understand demographic reasons of the past dynamics or to predict the population development in the future. Typical quantities of interest are the asymptotic population growth rate, the stable stage distribution, stage-specific reproductive values, generation time, net reproductive rate, growth rate sensitivities and elasticities. They can all be calculated for the matrix population model using well established methods (Caswell 2001), provided that information about all demographic rates in the transition matrix is available. If the demographic rates are known without error and the model does not include stochasticity, most of these quantities can be, and typically are, calculated from the transition matrix using matrix calculus (Caswell 2001). However, as soon as the demographic parameters are not known exactly, but subject to measurement or estimation error, or if environmental or demographic stochasticity needs to be included in the model, simulations are usually required for the estimation of these target quantities. Simulation also allows assessment of the precision of these quantities.

In this section we illustrate how the population growth rate and other important life history summaries are calculated classically. We start with the simplifying assumption that the demographic rates are constant over time (i.e., there is no environmental stochasticity), that the demographic rates are perfectly known (i.e. there is no estimation error) and that we do not account for the discrete nature of individuals (i.e. there is no demographic stochasticity). We will later relax these assumptions and show how to include environmental stochasticity, estimation errors and demographic stochasticity into our inferences from the transition matrix. Finally, we will introduce models that include density-dependence. We illustrate the computations with a female-based, stage-structured, pre-birth-pulse population model for the woodchat shrike that we have developed in chapter 3.2.3. (see Fig. 3.3.b). We assume that juvenile survival (s_j, survival from fledging until 1 year of age) is 0.3, adult survival (s_a, annual survival from 1 year of age onwards) is 0.55, productivity of 1-year old females (f_1, number of female fledglings per 1-year old female and year) is 1.3 and productivity of older females is f_a = 1.8 (these values are realistic for this species; see Lefranc and XXXXX.....). In section 3.4. we then illustrate how the same analyses can be carried out using Markov chain simulation (i.e., MCMC) methods. Methods for assessing demographic reason of population changes and for population projection into the future are treated more in details in chapters 8 and 9, respectively.

3.3.1. Analysis of a matrix projection model without stochasticity and parameter uncertainty

In the following we show the traditional calculation of the asymptotic population growth rate, the stable stage distribution and the sensitivities of the population growth rate to changes in the demographic rates. We could use for this R packages such as `popbio` (Stubben and Milligan 2007) or `demogR` (Jones 2007), or software such as Unified Life Models (ULM, Ferrière et al. 1996), but instead we show how these life history summaries can be calculated using standard R

functions. This has the advantage that the calculations become more transparent for you. We start by defining the demographic rates and the transition matrix **A**.

```
# Define the demographic rates
sj <- 0.3      # juvenile survival
sa <- 0.55     # adult survival
f1 <- 1.3      # number of female fledglings per 1-year old female and year
fa <- 1.8      # number of female fledglings per adult female and year

# Define the transition matrix A
A <- matrix(c(f1*sj, fa*sj, sa, sa), ncol = 2, byrow = TRUE)
dimnames(A) <- list(c("1y", "ad"), c("1y", "ad"))
A              # print the matrix
      1y   ad
1y  0.39 0.54
ad  0.55 0.55
```

Asymptotic population growth rate: The asymptotic population growth rate (usually symbolized with λ) is the rate at which the population described by matrix A grows when the stage distribution has become stable, i.e. after the initial *transient phase*. To experience what asymptotic and transient behaviour means, we can start with an arbitrary population vector, project the population for a number of years and calculate the annual growth rate of each stage class and of the complete population, and the actual stage distribution. The initial population vector is

```
Ni <- c(10,1)
```

thus, we assume an extremely skewed stage distribution for illustrative purpose. Then we project the population for the next 7 years, calculate in each year the actual growth rates of each stage class and of the complete population and the actual stage distribution and plot these quantities.

```
Ti <- 7
N <- matrix(NA, nrow = 2, ncol = Ti)
gr <- matrix(NA, nrow = 3, ncol = Ti-1)
sr <- numeric()
N[,1] <- Ni
for (t in 2:Ti){
   N[,t] <- A %*% N[,t-1]  # stage-specific population sizes
   gr[1,t-1] <- N[1,t] / N[1,t-1]
   gr[2,t-1] <- N[2,t] / N[2,t-1]
   gr[3,t-1] <- (N[1,t] + N[2,t]) / (N[1,t-1] + N[2,t-1])
   }
for (t in 1:Ti){
   sr[t] <- N[1,t] / (N[1,t] + N[2,t])
  }

# Plot with stage-specific population size
# Plit with stage-specific growth rates
# Plot with actual stage distribution

par(mfrow = c(1,3), las = 1, cex = 1.1)
plot(N[1,], type = "b", pch = 16, ylim = range(c(min(N), colSums(N))), axes =
FALSE, ylab = "Population size", xlab = "Time", col = "red")
points(N[2,], type = "b", pch = 16, col = "orange")
points(colSums(N), type = "b", pch = 16, col = "blue")
axis(1)
axis(2)
```

```
plot(x = (1:(Ti-1)) + 0.5, y = gr[1,], type = "b", pch = 16, ylim = range(gr),
xlim = c(1, Ti), axes = FALSE, ylab = "Population growth rate", xlab = "Time",
col = "red")
points(x = (1:(Ti-1)) + 0.5, y = gr[2,], type = "b", pch = 16, col = "orange")
points(x = (1:(Ti-1)) + 0.5, y = gr[3,], type = "b", pch = 16, col = "blue")
axis(1, at = 1:Ti, labels = 1:Ti)
axis(2)
abline(h = max(Re(eigen(A)$values)), lty = 2)

plot(sr, type = "b", pch = 16, ylab = "Proportion of stage classes", axes =
FALSE, col = "red", ylim = range(c(sr, 1-sr)))
points(1-sr, type = "b", pch = 16, col = "orange")
axis(1)
axis(2)
u <- which.max(Re(eigen(A)$values))
revec <- Re(eigen(A)$vectors[,u])
abline(h = revec[1]/sum(revec), col = "red", lty = 2)
abline(h = revec[2]/sum(revec), col = "orange", lty = 2)
legend("topright", legend = c("1y", "ad", "total"), bty = "n", pch = rep(16,
3), lty = rep(1, 3), col = c("red", "orange", "blue"))
```

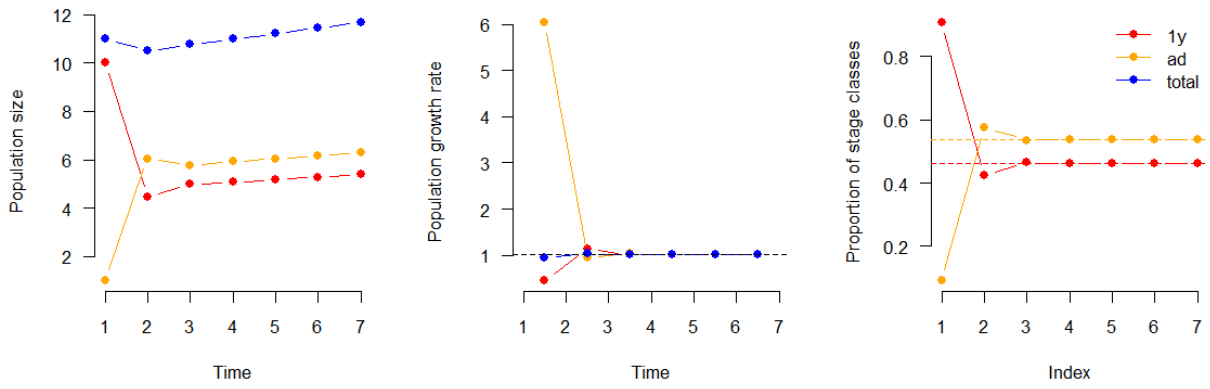


Figure 3-7 Development of stage-specific and total population sizes (a), stage-specific and total population growth rates (b) and stage distribution (c) calculated from the matrix projection model for woodchat shrikes assuming constant demographic rates and absence of demographic stochasticity. The transient phase takes only about 3 years, afterwards the asymptotic phase starts which is characterised by a stable stage distribution and constant and identical (across stage classes) population growth rates. The horizontal dashed lines show the asymptotic population growth rate and the stable stage distribution as obtained by the dominant eigenvalue and the right eigenvector of the transition matrix, respectively.

From Fig 3.7a it is obvious that the stage-specific population sizes are growing at the same rate from about the fourth year onwards. The stage-specific population growth rates are identical and they are identical to the growth rate of the total population size (Fig. 3.7b). The proportions of 1-year old individuals and of older individuals in the population change from year 1 to year 3, but remain stable after year 3 (Fig. 3.7c). Thus, the stage distribution is stable in this example from year 3 onwards, and the population growth rate remains also the same. Thus, after the short transient phase (from year 1 to year 3), the population shows asymptotic

behaviour. The growth rate in the latter phase is the asymptotic growth rate and the stable stage-distribution also refers to that phase. Many population statistics describe the *asymptotic properties* of population. As we will see later, population hardly ever experience asymptotic behaviour (they are never in a stable stage distribution), because populations get continuously "disturbed" by non constant demographic rates and demographic stochasticity. However, population statistics referring to the asymptotic phase are often very useful and good approximations, in particular for life cycles that reach a stable stage distribution very quickly, as the woodchat shrike example. It is important, though, that you understand what is meant by asymptotic and transient behaviour.

The length of the transient phase, i.e. the time to the moment when the population starts growing with the asymptotic rate, is related to the so-called *damping ratio*, which is the ratio of the largest (dominant) to the second largest eigenvalue (Caswell 2001, p. 95). The damping ratio is proportional to generation time, the shorter the generation time, the faster is convergence obtained (Ezard et al. 2010). To demonstrate that the transient behaviour can last longer, we look at the population of red kites, a long-lived bird of prey that starts to reproduce after 2 years. Survival increases with age until age 3, reproduction is low and restricted to females that are 3 years old or older and to few 2 years old individulas that start to reproduce already at this age. The life-cycle graph is composed of 4 stage classes (1-year old non-breeders, 2-year old non-breeders, 2-year old breeders, adult breeders) and we again use a pre-breeding census model. The life cycle graph of the red kite model is shown in chapter 6, where we construct an integrated population model for this species (Fig. 6.XY). The demographic rates, the Leslie (transition) matrix, the calculations and the figure code in appendix XY.

```
# Define the demographic rates
sj <- 0.45     # juvenile survival
s1 <-0.68      # survival of 1-year old individuals
sa <- 0.83     # adult survival (after 2 years old)
al <- 0.3      # prop. start breeding at age 2
f <- 0.83      # number of female fledglings per adult female and year

# Define the transition matrix A
A <- matrix(c(0, 0, f*sj, f*sj, s1*(1-al), 0, 0, 0, s1*al, 0, 0, 0, 0, sa, sa,
sa), ncol = 4, byrow = TRUE)


Ti <- 9
Ni <- c(10, 5, 5, 3)
N <- matrix(NA, nrow = 4, ncol = Ti)
gr <- matrix(NA, nrow = 5, ncol = Ti-1)
sr <- matrix(NA, nrow = 4, ncol = Ti)
N[,1] <- Ni
for (t in 2:Ti){
   N[,t] <- A %*% N[,t-1]  # stage-specific population sizes
   gr[1,t-1] <- N[1,t] / N[1,t-1]
   gr[2,t-1] <- N[2,t] / N[2,t-1]
   gr[3,t-1] <- N[3,t] / N[3,t-1]
   gr[4,t-1] <- N[4,t] / N[4,t-1]
   gr[5,t-1] <- (N[1,t] + N[2,t] + N[3,t] + N[4,t]) / (N[1,t-1] + N[2,t-1] +
N[3,t-1] + N[4,t-1])
   }
for (t in 1:Ti){
  sr[1,t] <- N[1,t] / (N[1,t] + N[2,t] + N[3,t] + N[4,t])
  sr[2,t] <- N[2,t] / (N[1,t] + N[2,t] + N[3,t] + N[4,t])
  sr[3,t] <- N[3,t] / (N[1,t] + N[2,t] + N[3,t] + N[4,t])
  sr[4,t] <- N[4,t] / (N[1,t] + N[2,t] + N[3,t] + N[4,t])
  }
```

```
co <- colorRampPalette(c("red", "orange"))(4)
par(mfrow = c(1,3), las = 1, cex = 1.1)
plot(N[1,], type = "b", pch = 16, ylim = range(c(min(N), colSums(N))), axes =
FALSE, ylab = "Population size", xlab = "Time", col = co[1])
points(N[2,], type = "b", pch = 16, col = co[2])
points(N[3,], type = "b", pch = 16, col = co[3])
points(N[4,], type = "b", pch = 16, col = co[4])
points(colSums(N), type = "b", pch = 16, col = "blue")
axis(1, at = 1:Ti)
axis(2)

plot(x = (1:(Ti-1)) + 0.5, y = gr[1,], type = "b", pch = 16, ylim = range(gr),
xlim = c(1, Ti), axes = FALSE, ylab = "Population growth rate", xlab = "Time",
col = co[1])
points(x = (1:(Ti-1)) + 0.5, y = gr[2,], type = "b", pch = 16, col = co[2])
points(x = (1:(Ti-1)) + 0.5, y = gr[3,], type = "b", pch = 16, col = co[3])
points(x = (1:(Ti-1)) + 0.5, y = gr[4,], type = "b", pch = 16, col = co[4])
points(x = (1:(Ti-1)) + 0.5, y = gr[5,], type = "b", pch = 16, col = "blue")
axis(1, at = 1:Ti)
axis(2)
abline(h = max(Re(eigen(A)$values)), lty = 2)


plot(sr[1,], type = "b", pch = 16, ylab = "Proportion of stage classes", axes
= FALSE, col = co[1], ylim = c(0, 1), xlab = "Time")
points(sr[2,], type = "b", pch = 16, col = co[2])
points(sr[3,], type = "b", pch = 16, col = co[3])
points(sr[4,], type = "b", pch = 16, col = co[4])
axis(1, at = 1:Ti)
axis(2)
u <- which.max(Re(eigen(A)$values))
revec <- Re(eigen(A)$vectors[,u])
abline(h = revec[1]/sum(revec), col = co[1], lty = 2)
abline(h = revec[2]/sum(revec), col = co[2], lty = 2)
abline(h = revec[3]/sum(revec), col = co[3], lty = 2)
abline(h = revec[4]/sum(revec), col = co[4], lty = 2)

legend("topright", legend = c("1y", "2yN", "2yB", "AdB", "total"), bty = "n",
pch = rep(16,5), lty = rep(1, 5), col = c(co, "blue"))
```

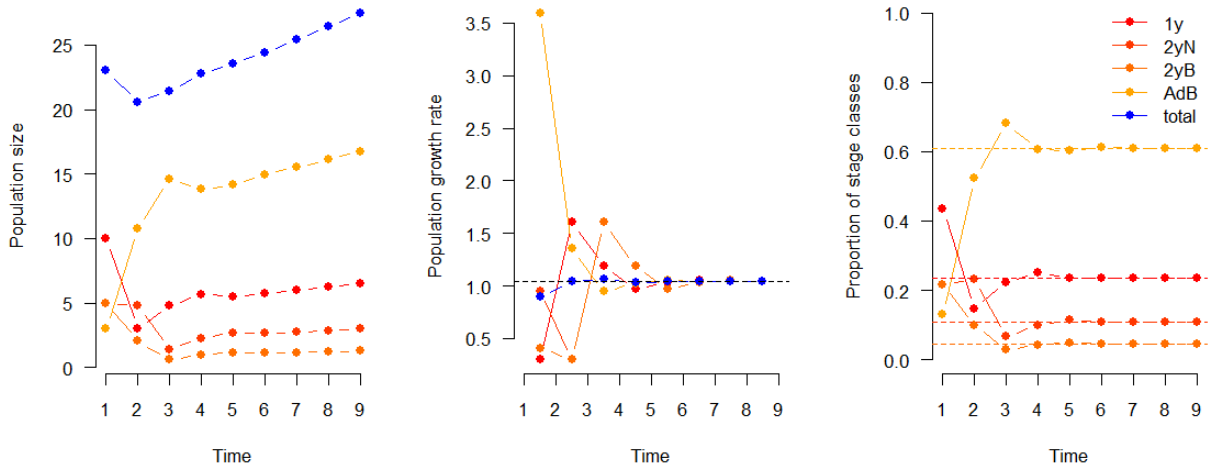


Figure 3-8 Development of stage-specific and total population sizes (a), stage-specific and total population growth rates (b) and stage distribution (c) calculated from the matrix projection model for red kites assuming constant demographic rates and absence of demographic stochasticity. The transient phase takes about 6 years, afterwards the asymptotic phase starts which is characterised by a stable stage distribution and constant and identical (across stage classes) population growth rates. The horizontal dashed lines show the asymptotic population growth rate and the stable stage distribution as obtained by the dominant eigenvalue and the right eigenvector of the transition matrix, respectively.

The asymptotic population growth rate is defined as the rate at which the population is growing annually after the stable stage distribution has been reached. Thus, starting from an arbitrary population vector, we could project the population for a sufficient number of years (e.g. *j*) and then calculate the asymptotic population growth rate as $\boldsymbol{N}_{j+1}/\boldsymbol{N}_j$. An alternative way is the application of linear algebra with which it can be shown that the asymptotic population growth rate is identical to the dominant (i.e., maximum) eigenvalue of **A** (Caswell 2001, chapter 4.4). In R we use the function `eigen` to get the desired quantity:

```
# Asymptotic population growth rate (lambda)
lambda <- max(Re(eigen(A)$values))
[1] 1.0208
```

λ is 1.0208, that is the population is growing by about 2.1% annually. For a better understanding, it may be helpful for you to execute this from the inside out (starting at `eigen(A)`) and inspecting all the intermediate results leading up to the value of `lambda`.

The calculation of the dominant eigenvalue needs the solution of the characteristic equation, which is

$$\det(\mathbf{A}-\lambda\mathbf{I})=0$$

where **A** is the transition matrix, **I** is an identity matrix (i.e. a square matrix with 1's on the diagonal and 0's at all other places) and det is the determinant. We insert transition matrix **A** and get

$$\det\left(\begin{bmatrix} f_1 s_j & f_a s_j \\ s_a & s_a \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = 0$$

After a bit of calculation we obtain

$\lambda^2 - \lambda\left(s_a + s_j f_1\right) + s_j s_a \left(f_1 - f_a\right) = 0$, and when we plug in the demographic rates we get

$\lambda^2 - 0.94\lambda - 0.0825 = 0$.

This is the characteristic equation of matrix **A**. Since **A** is a 2 by 2 matrix, the characteristic equation is quadratic in λ for which there are two solutions. The first is $\lambda_1 = -0.08081757$, and the second is $\lambda_2 = 1.02081757$. The dominant eigenvalue, i.e. λ_2, is the one that describes the asymptotic population growth rate. In simpler cases (e.g. if $f_1 = f_a$) it is possible to get algebraical solutions for the characteristic equation.

Stable stage distribution: When all demographic rates are constant most stage-structured populations that develop over a number of years will grow with the asymptotic population growth rate and converge to a stable stage distribution. We have experienced this behaviour by means of two examples (see Figs 3.7 and 3.8) and it is confirmed by theory (Cohen 1979). The strong ergodic theorem states that the long-term dynamic of a population is determined by the population growth rate, convergences to a stable stage structure and becomes independent of the initial stage structure if **A** is primitive (Caswell 2001, chapter 4.5). Most projection matrices are primitive, but in some cases they may not be. If **A** is imprimitive, the population cycles and if **A** is reducible the long-term dynamics depends on the initial conditions, thus it is not ergodic. Remember that a stable stage distribution means that the relative number of individuals in the different stages does not change anymore when the population grows for further years. Therefore, we can calculate the stable stage distribution from the stage-specific population sizes after the population has been projected for a sufficient number of times such that the stable stage distribution has been obtained. The stable stage distribution can also be calculated as the right eigenvector corresponding to the dominant eigenvalue of **A**. Thus, we first check which eigenvalue is dominant and then take the corresponding eigenvector:

```
u <- which.max(Re(eigen(A)$values))
[1] 1
revec <- Re(eigen(A)$vectors[,u])
[1] -0.6503047 -0.7596734
```

To get the stable stage distribution we standardise the eigenvector.

```
revec/sum(revec)

[1] 0.4612162 0.5387838
```

Thus, once the population growth has stabilized at its asymptotic value, 46.1% of the individuals will be 1-year old and 53.9% 2-years old or older. Remember that this stage structure refers to the time just before reproduction, because we have defined a population model corresponding to a pre-breeding census. Unlike the asymptotic population growth rate, the stable stage distribution is specific to the choice of census period (see exercise 3.1).

Stage-specific reproductive values: This measure expresses the relative contributions of each stage to the long-term population growth and thus, in a sense, is a measure of how "important" a stage is for population growth. Reproductive values can be decomposed into a component of current reproduction and a component of future reproduction (residual reproductive value) which is useful for studying life-history trade-offs (REF). They can be calculated as the left eigenvector corresponding to the dominant eigenvalue of transition matrix **A**. Often we scaled them so they sum to 1 to provide proportional reproductive values, or we scale it in such a way that the reproductive value of the youngest age class gets 1. The reproductive values of the older age classes can then be directly compared against that of the youngest age class.

```
u <- which.max(Re(eigen(A)$values))
levec <- Re(solve(eigen(A)$vectors)[u,])

levec/levec[1]
[1] 1.000000 1.159056
```

In our two-stage-class model the reproductive values of the younger age class is slighthly less than that of the oder age class, thus 1-year old woodchat shrikes contribute less to population growth than adult woodchat shrikes. An adult woodchat shrike is about 1.16 times more valuable than a 1-year old shrike for population growth.

Net reproductive rate: The net reproductive rate (R_0) is the expected, or mean, number of offspring by which a newborn shrike will be replaced by the end of its life. It measures the growth of the population *per generation*, not annually as does λ. One way to calculate R_0 is based on the Lotka equation as

$$R_0 = \sum_{i=1}^{\infty} l_i f_i$$

where l_i is the probability to survive from birth to stage class *i* and f_i is the productivity of stage class *i*. In our shrike example the net reproductive value is

$R_0 = s_j f_1 + s_j s_a f_a + s_j s_a^2 f_a + s_j s_a^3 f_a + \ldots + s_j s_a^\infty f_a = s_j \left(f_1 + f_a \sum_{i=1}^{\infty} s_a^i \right)$. In R we calculate this sum for 100 elements which is enough to get an accurate value.

```
i <- 1:100              # 100 as our approximation to infinity
R0 <- sj*f1+sj*fa*sum(sa^i)
[1] 1.05
```

Thus, each shrike is expected to be replaced by 1.05 conspecifics at the end of its life.

Generation time: This is often used to characterise the life history of a species for ranking them along the slow-fast gradient (Oli and Dobson 2003, Gaillard et al. 2005). There are several possible definitions of generation time (Caswell 2001, chapter 5.3.5.). The most common one is probably the mean age of mothers of offspring produced by a population at the stable stage distribution (this is called generation length in ULM (Legendre and Clobert 1995)). It is calculated as the sum over all stage classes as:

$$K = \sum_{i=1}^{\infty} i l_i f_i \lambda^{-i}$$

where l_i and f_i are the same as defined above and the sum is over 1 to infinity, but we will use a reasonably large number to approximate its upper bound. In the shrike example we get

$$K = s_j f_1 \lambda^{-1} + 2 s_j s_a f_a \lambda^{-2} + 3 s_j s_a^2 f_a \lambda^{-3} + \ldots = s_j \left(f_1 \lambda^{-1} + f_a \sum_{i=2}^{\infty} i s_a{}^{i-1} \lambda^{-i} \right)$$

which is calculated in R as

```
i <- 2:100
K <- sj*f1/lambda + sj*fa*sum(i*sa^(i-1)*lambda^(-i))
[1] 2.339834
```

Thus, the generation time is just a little over 2 years. Another definition of generation time is the time needed for a population to increase by the net reproductive rate (R_0). Hence,

$$GT = \frac{log(R_0)}{log(\lambda)}$$

```
GT <- log(R0) / log(lambda)
[1] 2.368012
```

Typicaly, these common measures of generation time are very similar (Caswell 2001, p. 129).

Sensitivity and elasticity: The population growth rate is a function of the demographic rates. As a measure of importance of each demographic rate we can compute the sensitivity and elasticity, which express the expected change of the growth rate when the value of a given demographic rate is varied by a certain (small) amount. Such analyses are often referred to as *perturbation analyses* (Caswell 2000; chapters 9 and 10, Caswell 2001). A typical application for a perturbation analysis is to predict population dynamics when future changes of demographic rates occur; this is a *prospective perturbation analysis*. For a species of conservation concern we aim at changeing, through appropriate management, the demographic rate that affects the population growth particularly strongly. Or for getting rid of an invasive species we need to know the most efficient strategy. Is population growth reduced most strongly when survival is decreased through culling or can we impact population growth more by lowering reproductive success, e.g. by destroying nesting sites? Related to such perturbation analyses are population viability analyses which will be treated in chapter XY. Another application of perturbation analyses is the assessment of how strongly each demographic rate has contributed to the population dynamics observed in the past. Such *restrospective perturbation analyses* are treated in more detail in chapter XY. For all perturbation analyses the sensitivities of the population growth rate to changes in demographic parameters play a central role. There are two commonly used such measures: sensitivity (*S*) and elasticity (*E*). Sensitivity measures the effect of *absolute* changes in the demographic rates (e.g., by 0.1) while elasticity measures effects of *relative*

changes (e.g., by 1%). Specifically, the sensitivity of the population growth rate (λ) to changes in element a_{ij} of projection matrix **A** is the first partial derivative of λ with respect to a_{ij}

$$S(a_{ij}) = \frac{\partial \lambda}{\partial a_{ij}}.$$

The elasticity is a scaled version of sensitivity

$$E(a_{ij}) = \frac{\partial \lambda}{\partial a_{ij}} \frac{a_{ij}}{\lambda}.$$

In both ratios, you can recognize the limit of a small change in the growth rate λ per small change in element a_{ij}. In *E*, the additinal term serves to scale the sensitivity such that we deal with relative changes in a_{ij}.

Empirical computation of *S* and *E* is based on the two eigenvectors – the sensitivity of the population growth rate to changes in element a_{ij} is the product of *i*th element of the stable stage distribution and the *j*th element of the reproductive value vector (Caswell 2001; chapter 9.1.). In R the sensitivity matrix is computed as

```
senmat <- levec %*% t(revec)
```

```
             [,1]      [,2]
[1,] 0.4273807 0.4992579
[2,] 0.4901804 0.5726193
```

and the elasticity matrix

```
elasmat <- senmat * A / lambda
```

```
           1y        ad
1y 0.1632794 0.2641013
ad 0.2641013 0.3085180
```

Typically, we don't want to know sensitivity or elasticity of the population growth rate to changes in an element of the project matrix directly (e.g., to element 1, which is fs_j), but rather to the changes of demographic rates of which the matrix elements are composed; the latter are called “lower level sensitivities/elasticities” (Caswell 2001; chapter 9.2.3.). To compute them we have to sum all elements of the sensitivity matrix and weigh them with the first partial derivative of the projection matrix with respect to the demographic rate. Thus, the sensitivity of the population growth rate to a small change in demographic rate *x* is

$$S(x) = \sum_{i,j} \frac{\partial \lambda}{\partial a_{ij}} \frac{\partial a_{ij}}{\partial x},$$

and the elasticity is

$$E(x) = \frac{x}{\lambda} \sum_{i,j} \frac{\partial \lambda}{\partial a_{ij}} \frac{\partial a_{ij}}{\partial x}.$$

Let’s use R to calculate growth rate sensitivity and elasticity to changes in juvenile survival in our shrike example ($S(s_j)$ and $E(s_j)$). We first have to calculate the first partial derivatives of the projection matrix with respect to juvenile survival,

$$\frac{\partial \begin{bmatrix} f_1 s_j & f_a s_j \\ s_a & s_a \end{bmatrix}}{\partial s_j} = \begin{bmatrix} f_1 & f_a \\ 0 & 0 \end{bmatrix}.$$

Next, the matrix of derivatives is multiplied with the sensitivity matrix and the elements of the resulting matrix are summed up.

```
derivmat <- matrix(c(f1, fa, 0, 0), ncol = 2, byrow = TRUE)
ssj <- sum(senmat * derivmat)     # Sensitivity
esj <- sj / lambda * ssj          # Elasticity
ssj; esj

[1] 1.454259
[1] 0.4273807
```

Finally, we can experience sensitivities graphically. We change the demographic rate whose growth rate sensitivity is to be calculated by small positive and negative values and calculate the population growth rates using each of them. The plot the the resulting population growth rates against the demographic rate shows the sensitivity: it is the slope of this relationship (Fig. XY).

3.3.2. Analysis of a projection matrix model without stochasticity, but with parameter uncertainty

Demographic rates are very rarely known perfectly, but instead virtually always need to be estimated. This means that they are known only up to some degree of uncertainty and consequently, this uncertainty ought to be propagated into any function of the demographic rates, such as all the population summaries that we just computed in the preceding section. Several approaches are possible to achieve this uncertainty, or error, propagation, including approximations such as the Taylor series expansion, also called the delta rule (Powell 2007, Caswell 2001, chapter 12), and various kinds of simulations. The latter are most certainly the most prominent and powerful methods and include bootstrapping (Brault and Caswell 1993, Caswell 2001, chapter 12) and Monte Carlo simulations (Caswell et al. 1998). Here we focus on Monte Carlo simulations, because they are easy to perform in general and because they share similarities to the Bayesian MCMC methods which we also use for parameter estimation throughout this book.

The principle of Monte Carlo simulation is fairly easy. The only difficulty is that the uncertainty associated with a demographic parameter must be described with a statistical distribution. The Monte Carlo simulation works as follows:

1. Generate a value of each demographic rate from the specified distributions.
2. Use those values to calculate the asymptotic population growth rate and/or another life history summary.
3. Repeat steps 1-2 many times and save the results each time.
4. Calculate the mean and the standard deviation of the quantities of interest as a point estimate and a quantity analogous to a standard error.

Let's here assume that we have estimated age-specific survival probabilities of the woodchat shrikes from a capture-recapture data set. We assume the same means as before, s_j = 0.3 and s_a = 0.55, and standard errors (or, in a Bayesian analysis, posterior standard deviations) of SD(s_j) = 0.05 for juvenile and SD(s_a) = 0.03 for adult survival. When we simulate survival probabilities from a distribution we have to respect the constraint that the simulated values

must be in the permissible range between 0 and 1. If the means are far away from these bounds and the SD's small, we can sample a normal distribution with these values for the mean and SD directly. But a safer option is to use of a normal distribution, but with a transformed scale (e.g., the logit). It requires that the mean and the standard deviation are transformed to this scale. In the case of the logit scale, the mean is $\text{logit}(x) = \log\left(\frac{x}{1-x}\right)$, while the standard deviation is $SD(\text{logit}(x)) \cong \frac{SD(x)}{x(1-x)}$. A third possibility that does not need a transformation is the use of the beta distribution. The beta distribution is specified by two scale parameters that are calculated from the known mean and standard deviations. This is done with the following function:

```
# Function to translate mean and sd of survival into parameters of a beta
distribution
beta.params <- function(x_bar, sd.x){
   u <- x_bar*(1-x_bar)/sd.x^2-1
   alpha <- x_bar*u
   beta <- (1-x_bar)*u
   return(list(alpha = alpha, beta = beta))
   }
```

For fecundity we now also assume that we have estimates; the means are as before f_1 = 1.3 and f_a = 1.8, and each of them have a SE of SD(f_1) = 0.3 and SD(f_a) = 0.1, respectively. Fecundity has no upper bound, but must be positive. A possible distribution to generate values is therefore the log-Normal distribution. If the standard deviation is small enough relative to the point estimate, it is also possible to use the Normal distribution on the natural scale. Since the standard deviations are small in our example, there is an extremely low risk that negative values are generated and thus no transformation (no link function) is needed here.

For illustration, we now compute several life-history summaries such as the asymptotic population growth rate, the stable stage distribution, lower-level sensitivities, net reproductive rate and generation time. The following bits of R code perform Monte Carlo simulations to calculate the precision of the asymptotic population growth rate and the target life-history summaries.

```
# Define mean and SD of the demographic parameters
mean.sj <- 0.3        # Point estimate of juv. survival
sd.sj.e <- 0.03       # Uncertainty of juv. survival expressed as SD on natural
scale
mean.sa <- 0.55       # Point estimate of ad. survival
sd.sa.e <- 0.015      # Uncertainty of ad. survival expressed as SD on natural
scale
mean.f1 <- 1.3         # Point estimate of fecundity of 1y females
sd.f1.e <- 0.3         # Uncertainty of fecundity expressed as SD on natural
scale
mean.fa <- 1.8         # Point estimate of fecundity of ad females
sd.fa.e <- 0.1         # Uncertainty of fecundity expressed as SD on natural
scale

# Define number of simulations, vectors and matrices to store results
nsim <- 10000
lambda <- R0 <- GT <- numeric()
stable.stage <- matrix(NA, ncol = 2, nrow = nsim)
sensitivity <- matrix(NA, ncol = 3, nrow = nsim)

# Generate demographic values from beta and normal distributions
```

```
sj.sim <- rbeta(nsim, beta.params(mean.sj, sd.sj.e)$alpha,
beta.params(mean.sj, sd.sj.e)$beta)
sa.sim <- rbeta(nsim, beta.params(mean.sa, sd.sa.e)$alpha,
beta.params(mean.sa, sd.sa.e)$beta)
f1.sim <- rnorm(nsim, mean.f1, sd.f1.e)
fa.sim <- rnorm(nsim, mean.fa, sd.fa.e)

# Perform Monte Carlo simulation
for (s in 1:nsim){
   if(s %% 1000 == 0) {cat(paste("*** Simrep", s, "***\n"))}  # Counter
   # Projection matrix
   A   <-   matrix(c(sj.sim[s]*f1.sim[s],   sj.sim[s]*fa.sim[s],   sa.sim[s],
sa.sim[s]), ncol = 2, byrow = TRUE)
   # Asymptotic population growth rate
   lambda[s] <- max(Re(eigen(A)$values))

   # Stable stage distribution
   u <- which.max(Re(eigen(A)$values))
   revec <- Re(eigen(A)$vectors[,u])
   stable.stage[s,] <- revec/sum(revec)

   # Lower level sensitivities
   levec <- Re(solve(eigen(A)$vectors)[u,])
   senmat <- levec%*%t(revec)
   derivmat <- matrix(c(f1.sim[s], fa.sim[s], 0, 0), ncol = 2, byrow = TRUE)
   sensitivity[s,1] <- sum(senmat * derivmat)
   derivmat <- matrix(c(sj.sim[s], sj.sim[s], 0, 0), ncol = 2, byrow = TRUE)
   sensitivity[s,2] <- sum(senmat * derivmat)
   derivmat <- matrix(c(0, 0, 1, 1), ncol = 2, byrow = TRUE)
   sensitivity[s,3] <- sum(senmat * derivmat)

   # Net reproductive rate
   i <- 1:100
   R0[s] <- sj.sim[s]*f1.sim[s]+sj.sim[s]*fa.sim[s]*sum(sa.sim[s]^i)
   # Generation time
   GT[s] <- log(R0[s]) / log(lambda[s])
   }

# Summaries of the quantities of interest
mean(lambda)
[1] 1.022953
sd(lambda)
[1] 0.06237772

mean(stable.stage[,1])
[1] 0.460675
sd(stable.stage[,1])
[1] 0.03198174
mean(stable.stage[,2])
[1] 0.539325
sd(stable.stage[,2])
[1] 0.03198174

mean(sensitivity[,1])
[1] 1.465529
sd(sensitivity[,1])
[1] 0.1905539
mean(sensitivity[,2])
[1] 0.2783583
sd(sensitivity[,2])
[1] 0.03095104
mean(sensitivity[,3])
[1] 1.060417
```

```
sd(sensitivity[,3])
[1] 0.03671012

mean(R0)
[1] 1.053344
sd(R0)
[1] 0.148235

mean(GT)
[1] 2.382681
sd(GT)
[1] 0.188045


# Produce figure 3.9.
par(mfrow = c(2, 2), mar = c(5, 4, 3, 0), las = 1, cex = 1.1)
a <- hist(lambda, nclass = 50, col = "dodgerblue", main = "", xlab =
expression(paste("Asymptotic population growth rate (", lambda, ")")), prob =
TRUE)
text(x = a$mids[2], y = max(a$density), "a")
par(mar = c(5, 2, 3, 2))
a <- hist(stable.stage[,1], nclass = 50, col = "dodgerblue", main = "", xlab =
"Proportion of 1-year old individuals", prob = TRUE)
text(x = a$mids[2], y = max(a$density), "b")
par(mar = c(5, 4, 3, 0))
a <- hist(sensitivity[,1], nclass = 50, col = "dodgerblue", main = "", xlab =
expression(paste("Sensitivity of ", lambda, " to variation of juvenile
survival")), prob = TRUE)
text(x = a$mids[2], y = max(a$density), "c")
par(mar = c(5, 2, 3, 2))
a <- hist(GT, nclass = 30, col = "dodgerblue", main = "", xlab = "Generation
time", prob = TRUE)
text(x = a$mids[2], y = max(a$density), "d")
```

We note that there is sampling error associated with these simulation-based solutions. That is, unless we all initialize the random number generators (using `set.seed(x)` with some common value for x) you will not get the exact same estimates as we do. Nevertheless, 10000 samples are usually enough to get accurate estimates of the mean and the standard deviation of the life history summaries, such that the sampling variability can be ignored. However, if you want more accuracy, simply draw more simulation replicates.

We see that the frequency distribution of these quantities is not strongly skewed (Fig. 3.9.) and thus the SD is an appropriate metric to describe uncertainty. The precision of the estimates does not change substantially when the number of samples is increased.

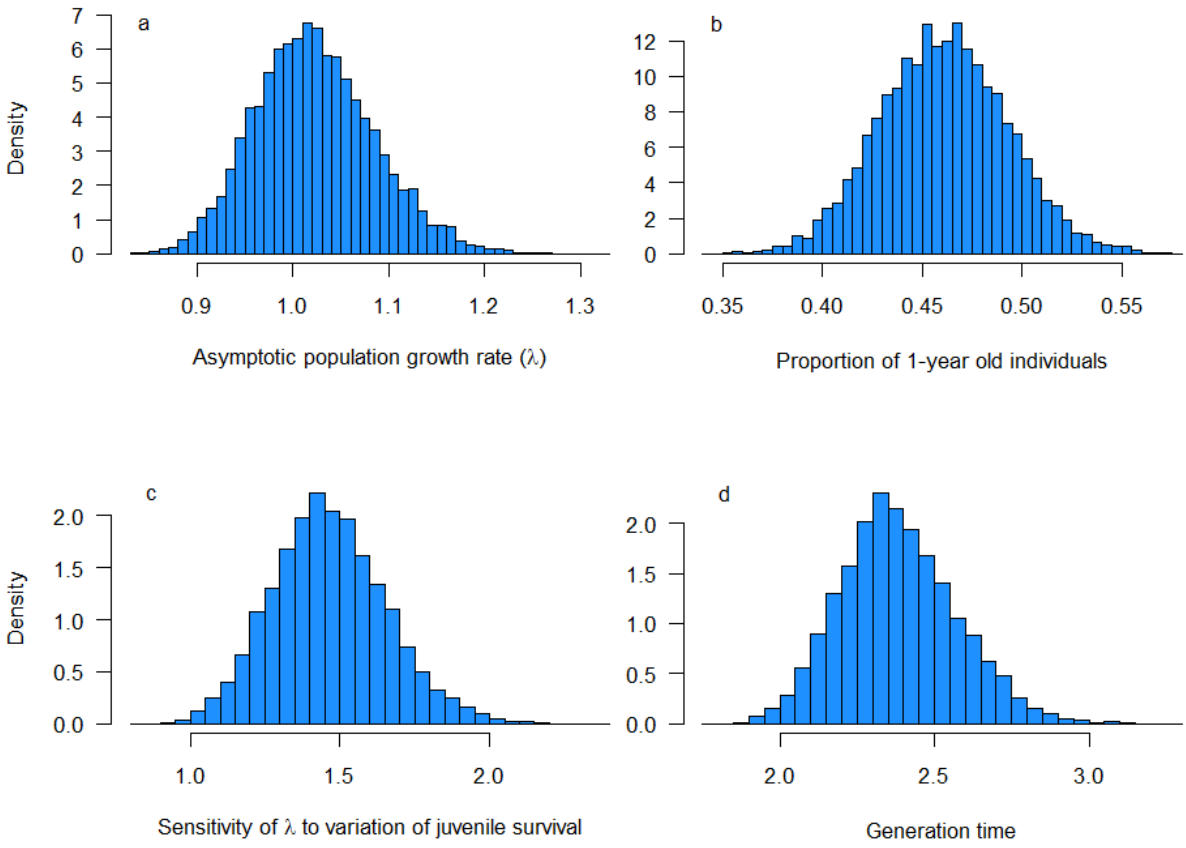


Figure 3.9 Frequency distribution of the asymptotic population growth rate (a), of the proportion of 1-year old individuals (b), of growth rate sensitivity to variation in juvenile survival (c) and of generation time (d) as obtained from 10,000 Monte Carlo samples.

3.3.3. Analysis of a projection matrix model with environmental stochasticity, but without parameter uncertainty

In reality, demographic rates are rarely constant over time, but vary from year to year, something we call environmental stochasticity. The magnitude of the temporal variation of demographic rates can be substantial (Gaillard et al. 2000, Altwegg et al. 2007), and the calculation of the population growth rate and of life history summaries needs to take this variation into account. All calculations so far assumed constant demographic rates; they cannot be used when demographic rates are variable, because they would provide biased values. Projection of the population is a good way to calculate population growth rate and life history summaries when environmental stochasticity occurs (Caswell 2001, Morris and Doak 2002) and we demonstrate this here.

We start with the calculation of the so-called stochastic population growth rate. The principle that needs to be performed is fairly easy: the population size is projected for a large number of years and the annual population growth rates are calculated. In a homogenous stochastic environment where the pattern of variation and covariation are stationary, the log population size becomes asymptotically normal and the average growth rate converges to a

fixed value (Tuljapurkar and Orzack 1980). This value, the stochastic population growth rate, is calculated as the geometric mean of the annual growth rates. We must project the population for a very large number of years; the precision of the stochastic growth rate increases with the number of years. There are two challenges for the simulation. First, how can we get demographic rates for many years? We rarely have a demographic study that is running for, say, 1,000 years. Yet, we may have estimates of the mean and of the temporal variability of demographic rates that we have obtained by models with temporal random effects (). This allows the generation of as many values of demographic rates as we like. Importantly, when temporal random effects are estimated, sampling variation is eliminated (Gould and Nichols 1998). For the calculation of the stochastic population growth rates we have to include process variance (i.e. the temporal variability) only that is not contaminated with sampling variability. If the demographic rates are temporally correlated (e.g. becaue in a favourable year both juvenile and adult survival are enhanced), the temporal correlation should be estimated (Schaub et al. 2013, Link and Barker 2005) and included in the projection, because such correlations can have a strong impact on population growth. Alternatively we may have annual estimates of demographic rates from a decent number of years. To create values for more years, we can sample from the observed estimates with replacement. Sampling can be set up such that correlations among demographic rates are accounted for (if we sample years). In both approaches we assume that the conditions that we have observed are typical for the future, i.a., that we project a process that is stationary both in terms of its mean and its variance.

To project a population over time, we must initialize it at some values and a second challenge is therefore that the stage-specific population sizes are typically unknown. Yet, we can define any arbitrary stage-specific population vector, provided that its sum is larger than zero. As in any Markov chain, the growth of the population will be influenced by the choice of the initial stage-specific population sizes. However, in the long-term the population trajectory will become independent from these initial values (Caswell 2001, chapter 14.2.). Indeed, if the environmental stochastic process is homogenuous (constant mean and variability), the stage structure of the population converges to a fixed stationary distribution. To get rid of the potential impact of the choice of the population vector in the first year, the first growth rates can be discarded for the calculation of the mean as a sort of "burnin" of this Markov chain.

Assume for now that we have generated a large number (T) of samples of demographic rates ($s_{j,t}$, $s_{a,t}$, $f_{1,t}$, $f_{a,t}$) from which we construct year-specific projection matrices $\mathbf{A}_t$, and that the stage-specific population vector in year 1 is $\boldsymbol{N}_1$. We also assume here that the different demographic rates vary independently from each over over time, thus that temporal covariances are zero. The stage-specific population sizes in year t {t = 1…T} is then calculated as

$$\mathbf{N}_{t+1} = \mathbf{A}_t\mathbf{N}_t$$

Annual population growth rates are calculated from the population vectors. The annual growth rates in our shrike example with two age classes are calculated as

$$\lambda_t = \frac{N_{1,t+1} + N_{2,t+1}}{N_{1,t} + N_{2,t}} .$$

For ease of calculation the growth rate are usually expressed on the log scale ($r_t = \log(\lambda_t)$). Thus, the annual growth rates on the log scale are in our example

$$r_t = \log\left(N_{1,t+1} + N_{2,t+1}\right) - \log\left(N_{1,t} + N_{2,t}\right) .$$

The stochastic population growth rate is then calculated either as the geometric mean of the annual growth rates (λ_t) or as the arithmetic mean of the annual growth rates on the log

scale (r_t). Typically, the first *u* samples of the chain of annual growth rates are discarded to prevent the estimates from being influenced by the arbitrary choice of initial values:

$$\lambda_s = \left(\prod_{t=u}^{T} \lambda_t \right)^{\frac{1}{T-u}} \text{ or } r = \frac{1}{T-u} \sum_{t=u}^{T} r_t \text{, while } r = \log(\lambda_s).$$

The following R code calculates the stochastic population growth rate on the log scale (*r*).

```
# Define mean and temporal variability (SD) of the demographic parameters
mean.sj <- 0.3          # Mean juvenile survival (probability scale)
sd.sj.t <- 0.25         # Temporal variability on the logit scale
mean.sa <- 0.55         # Mean adult survival (probability scale)
sd.sa.t <- 0.07         # Temporal variability on the logit scale
mean.f1 <- 1.3          # Mean fecundity of 1y old females
sd.f1.t <- 0.3          # Temporal variability on the natural scale
mean.fa <- 1.8          # Mean fecundity of adult females
sd.fa.t <- 0.3          # Temporal variability on the natural scale

# Define the number of years with predictions and the discard
T <- 10000              # Length of Markov chain
u <- 100                # Length of burnin period
```

Models with temporal random effects usually use a Normal distribution applied on a transformed scale such as the logit for survival or the log for for fecundity. Therefore, any resulting estimate of the magnitude of temporal variability refers to the chosen scale and to generate values it is advisable to use also the Normal distribution. Since all demographic rates are generated independently, we assume that they vary independently from one another over time, or more technically, that the process covariances are zero. It is possible to relax this assumption when estimates of the temporal covariances are available (see chapter XY). The following code performs the desired simulations:

```
# Generate demographic values from normal distributions
sj <- plogis(rnorm(T, qlogis(mean.sj), sd.sj.t))
sa <- plogis(rnorm(T, qlogis(mean.sa), sd.sa.t))
f1 <- rnorm(T, mean.f1, sd.f1.t)
fa <- rnorm(T, mean.fa, sd.fa.t)

# Define population matrix and initial stage-specific population sizes
N <- matrix(NA, nrow = 2, ncol = T+1)
N[,1] <- c(1, 1)

# Project population forwards
r <- numeric()
for (t in 1:T){
   if(t %% 1000 == 0) {cat(paste("*** Simrep", t, "***\n")) }  # Counter
   A <- matrix(c(sj[t]*f1[t], sj[t]*fa[t], sa[t], sa[t]), ncol = 2, byrow =
TRUE)
   N[,t+1] <- A%*%N[,t]
   r[t] <- log(sum(N[,t+1])) - log(sum(N[,t]))  # Annual population growth
rate
   N[,t+1] <- N[,t+1]/sum(N[,t+1])       # Scale N to avoid numerical overflow
   }
```

The mean of the annual growth rates (after the *u* burnin samples) is an estimate of the stochastic population growth rate.

```
mean(r[u:T])
[1] 0.01863444
```

As always when using simulation, remember that you will obtain slightly different solutions depending on the initial value of your random numbers generator, hence, your solution will differ slightly from ours. We could eliminate this effect by either using chains of "infinite" length or else using a seed.

As seen above, we can also express the stochastic population growth rate on the natural scale for lambda, which is

```
exp(mean(r[u:T]))
[1] 1.018809
```

The stochastic population growth rate is smaller than the asymptotic population growth rate, although the means of the demographic rates were exactly the same. The only difference was that the demographic rates were temporally variable. This finding, that the stochastic population growth rate is smaller than the asymptotic (deterministic) population growth rate, is not just specific to our example, but is a general result. Theoretical proofs can be found in Tuljapurkar and Orzack (1980).

The precision of the stochastic population growth rate can be evaluated using Monte Carlo simulation. It increases with the number of projected years (T) (see exercise XY). But as we can make the precision as small as we like, it is not generally a very meaningful quantity, because it doesn't say us anything about the study population.

The sensitivity and elasticity of the stochastic population growth rate to changes in the demographic parameters are also calculated by simulations (Caswell 2001, chapter 14.4., Morris and Doak 2002). The population is projected twice for a large number of years, first using the demographic rates as they were estimated and second where one demographic rate is changed by a small amount. Each time the stochastic growth rate is calculated and the difference between the two growth rates tells us how sensible it is to the change in the target demographic rate. Specifically, we change the demographic parameter (a_j) whose growth rate sensitivity we want to calculate by a small amount ($a_j^* = a_j - \Delta$, where Δ is e.g. 0.001) and calculate the stochastic population growth rates (r and r^*) one time using a_j and one time using a_j^*. The sensitivity is then the ratio of the difference of the two stochastic population growth rates (on the natural scale) to the change in the demographic rate ($S(a_j) = \left(exp(r^*) - exp(r)\right) / \Delta$).To get the elasticity we need to scale the sensitivity: $E(a_j) = S(a_j) \times a_j / exp(r)$. The following code calculates the sensitivity and elasticity of the stochastic population growth rate to changes in juvenile survival.

```
# Generate demographic values from normal distributions
sj <- plogis(rnorm(T, qlogis(mean.sj), sd.sj.t))
sa <- plogis(rnorm(T, qlogis(mean.sa), sd.sa.t))
f1 <- rnorm(T, mean.f1, sd.f1.t)
fa <- rnorm(T, mean.fa, sd.fa.t)

# Define population matrix and initial stage-specific population sizes
N <- N.star <- matrix(NA, nrow = 2, ncol = T+1)
N[,1] <- N.star[,1] <- c(1, 1)

# Project population and calculate stochastic population growth rate
delta <- 0.001          # Magnitude of perturbation (= "small change")
r <- r.star <- numeric()
for (t in 1:T){
```

```
   if(t %% 1000 == 0) {cat(paste("*** Simrep", t, "***\n")) }  # Counter
   # Projection using sj
   A <- matrix(c(sj[t]*f1[t], sj[t]*fa[t], sa[t], sa[t]), ncol = 2, byrow =
TRUE)
   N[,t+1] <- A%*%N[,t]
   r[t] <- log(sum(N[,t+1])) - log(sum(N[,t]))  # Annual population growth
rate
   N[,t+1] <- N[,t+1]/sum(N[,t+1])       # Scale N to avoid numerical overflow

   # Projection using sj.star
   A.star <- matrix(c((sj[t]+delta)*f1[t], (sj[t]+delta)*fa[t], sa[t], sa[t]),
ncol = 2, byrow = TRUE)
   N.star[,t+1] <- A.star%*%N.star[,t]
   r.star[t] <- log(sum(N.star[,t+1])) - log(sum(N.star[,t]))  # Annual
population growth rate
   N.star[,t+1] <- N.star[,t+1]/sum(N.star[,t+1])       # Scale N.star to
avoid numerical overflow
   }

# Compute stochastic sensitivity (juvenile survival)
(exp(mean(r.star[u:T])) - exp(mean(r[u:T]))) / delta
[1] 1.448877

# Compute stochastic elasticity (juvenile survival)
(exp(mean(r.star[u:T])) - exp(mean(r[u:T]))) / delta * mean.sj /
exp(mean(r[u:T]))
[1] 0.4266363
```

Sensitivity and elasticity of the asymptotic and of the stochastic growth rates to changes in juvenile survival are very similar. Unless the temporal variability becomes very large, the similarity of deterministic and stochastic growth rate sensitivities is a common observation (Altwegg et al. 2007) and not specific to our example.

3.3.4. Analysis of a projection matrix model with demographic stochasticity, but without parameter uncertainty

Demographic stochasticity originates from random variation in the realized number of births and deaths in a population caused by the discrete nature of individuals. A jaguar can have 2 or 3 kittens, but not 2.5. In other words, demographic stochasticity can be described as the variance of a binomial or a Poisson random variable. To illustrate how demographic stochasticity works, a small example does help. We can ask how many of 10 woodchat shrikes survive until the next year, given the survival probability of s_a = 0.55. To address this question we can conduct 10 coins flips with a coin that produces heads in 55% of all cases, and count how many heads (= survival events) we obtain. Since such a loaded coin is difficult to get, the application of the binomial random number generator in R is more convenient:

```
rbinom(n = 1, size = 10, prob = 0.55)
[1] 4
```

Four individuals have survived in this particular realization of the chance experiment. When we repeat this 20 times, we get quite variable numbers of survivors (and you get a different set of numbers, remember):

```
rbinom(n = 20, size = 10, prob = 0.55)
[1] 4 6 7 6 7 4 4 7 9 3 5 5 6 6 1 6 2 6 1 6
```

The numbers of survivors now varies between 1 and 9, the most frequently got number is 6. The expected value of the experiment is 5.5 (= 10 * 0.55), but we never actually obtain this number of survivors, because it is impossible that five and a half shrikes are all alive. Yet, if we ignore demographic stochasticity, we conduct calculations in a population model with the expected value of this random variable, i.e., with 5.5 individuals. We also note that the number of surviving individuals varies between trials, although the survival probability was the same in the 20 trials. Put in a different way, binomial counts with sample size 20 and success probability 0.55 naturally vary from trial to trial.

Demographic stochasticity (i.e., the difference between the realized and the expected value of a discrete-valued random variable) is present in all populations, but it is relevant for population modelling particularly when the population size is small (Lande 1993). We can demonstrate this again using the example from before. We now repeate the generation of surviving individuals 1000 times and always calculate the *realized* proportion of surviving birds, i.e. the ratio of the number of surviving shrikes and the initial number of shrikes (10). We then repeat this experiment with progressively larger population sizes. Finally, we plot frequency distributions of the realized survival probabilities (Fig. 3.10).

```
size <- c(10, 25, 50, 75, 100, 250, 500, 750, 1000, 5000, 10000)
s <- matrix(NA, nrow = 1000, ncol = length(size))
for (i in 1:length(size)){
   s[,i] <- rbinom(n = 1000, size = size[i], prob = 0.55) / size[i]
   }
boxplot(s, ylab = "Realized proportion of surviving birds", xlab = "Population
size (number of females)", names = size, outline = FALSE, col = "dodgerblue",
las  = 1)
abline(h = 0.55, col = "red", lwd = 1)
```

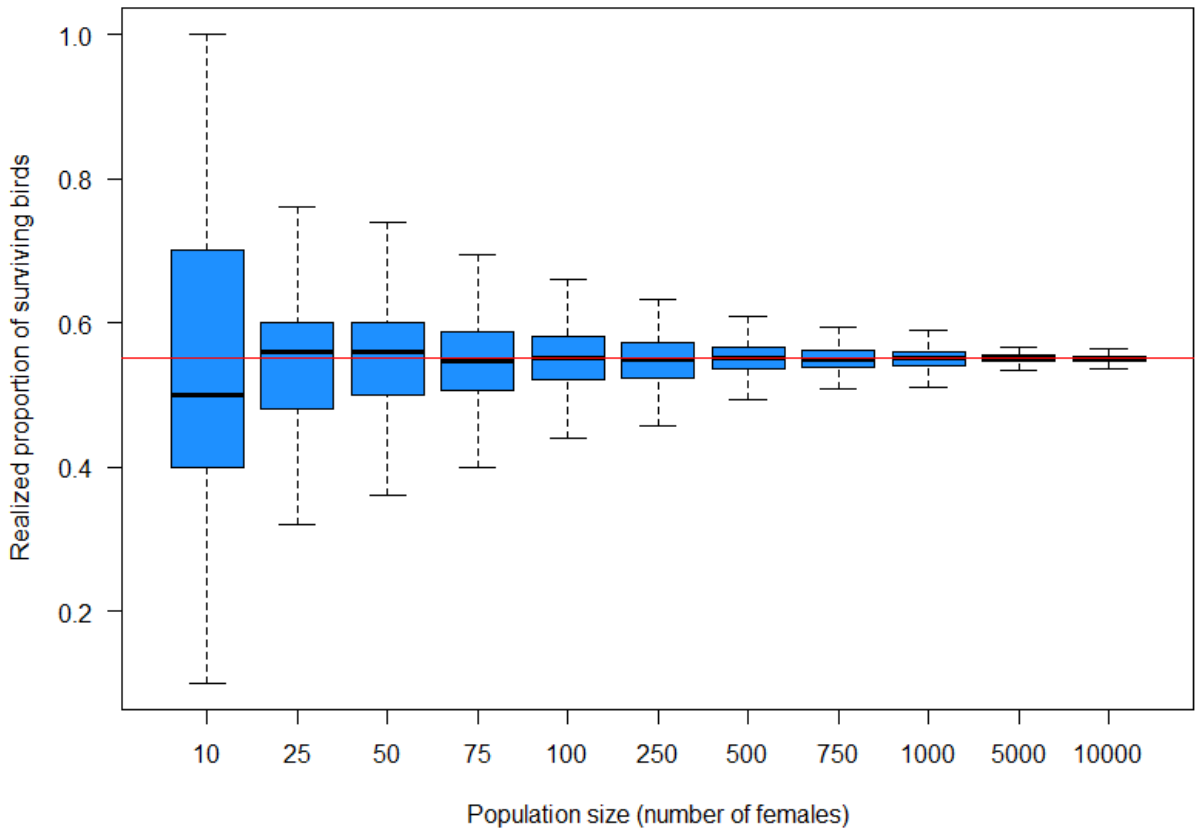


Figure 3.10 Boxplot of realized proportion of surviving birds, generated from 1000 binomial trials when the survival probability is 0.55 (i.e. success probability) and population size (i.e. the binomial total) increases from 10 to 10000. We see that smaller populations are much more affected by demographic stochasticity.

Realized survival is much more variable when the population consists a low number than a large number of shirkes, although the expected survival (i.e., the survival probability) was exactly the same in all cases. Smaller populations will therefore experience stronger temporal variability than larger populations, even if the underlying parameter governing the survival process (= survival probability) is identical. This applies not only to survival, but to all demographic processes that naturally correspond to discrete-valued random variables, and as a result also to the population growth rate: smaller populations vary more in size than larger populations.

In general, to include demographic stochasticity in a population model, we must simulate the demographic performance of each individual. This requires the application of appropriate statistical distibutions for discrete random variables. We have already seen that the binomial distribution is an appropriate choice for the survival process: out of N individuals that each survive with probability s, we can never observe more survivors than N and on average will observe Ns survivors. For productivity, the Poisson distribution is our first natural choice, although it has sometimes been found to be too variable (Ridout and Besbeas 2004). We will see more on possible statistical distributions for demographic processes later in the book. Another requirement for the analysis of a matrix population model with demographic

stochasticity is the definition of the stage-specific population vector in the first year. Since the "magnitude" of demographic stochasticity depends on the population size, the size of this vector now matters for the trajectory of a population affected by demographic stochasticity.

A final issue before we can start with simulations is the possible extinction of a population during a simulation. When a population is small and we account for the discrete nature of individuals, it can happen at some time step that no individual survives and no individual is recruited. The population size then drops to zero and the population goes extinct. In contrast, in the absence of demographic stochasticity a population can never go extinct. Population size may then get very, very small in the long term, but never *exactly* zero. Mathematically, this means that we do not need to bother about extinction in a population model without demographic stochasticity, because the population model is still working and the population growth rate is always defined. By contrast, as soon as the population size becomes zero (which can only happen in a model with demographic stochasticity), the population will remain at this size of zero, provided that there is no immigration. Consequently, the population growth rate is no longer defined (because, mathematically, we have a division by zero or try to take the log of zero). When we model a population including demographic stochasticity we must therefore account for the possibility that the population goes extinct and restrict the calculation of the population growth rate and other life-history summaries to that part of the population trajectory before extinction. The study of extinction is an important field in conservation biology (Beissinger 2002) and we will see more about this topic in chapter 10, where we use an integrated population model to estimate extinction probability and related parameters. But for now it is only important that we consider the possibility of population extinction in the calculation of the population growth rate.

We now know all we need to know to simulate a population with demographic stochasticity and can therefore start with an illustration to show how such a population evolves over many years. The algorithm is the following:

1. Define the stage-specific number of individuals in the first year
2. Loop over time: determine the number of individuals in each year using appropriate probability distributions for discrete random variables, which reflect the demographic processes in each year
3. Calculate the stochastic population growth rate in each year

The following R code performs these calculations. We assume that a population consists of 20 females in year 1, of which 10 are 1-year old and 10 are 2-years or older. Moreover, we use a single probability distribution (the Poisson) for the recruitment process. The recruitment process really consists of two distinct demographic processes: the production of fledglings (for which a Poisson looks adequate) and the survival of these fledglings until the next year (for which we naturally adopt a Binomial). Hence, it would also be possible to explicitly model these two processes. Here we opt for the simpler option, which is to model their product by a Poisson distribution. Nevertheless, we note that we will show examples later in the book for the former (chapter XY).

```
# Define mean of the demographic parameters
mean.sj <- 0.3
mean.sa <- 0.55
mean.f1 <- 1.3
mean.fa <- 1.8

# Define the number of years over which we let the population evolve
T <- 200
```

```
# Define population matrix and initial stage-specific population sizes
N <- matrix(NA, nrow = 2, ncol = T+1)
N[,1] <- c(10, 10)

# Project population
r <- numeric()
for (t in 1:T){
   N[1,t+1] <- rpois(1, mean.sj * mean.f1 * N[1,t] + mean.sj * mean.fa *
N[2,t])
   N[2,t+1] <- rbinom(1, sum(N[,t]), mean.sa)
   if (sum(N[,t+1]) == 0) break     # stop calculation if population extinct
   r[t] <- log(sum(N[,t+1])) - log(sum(N[,t]))
   }
mean(r)                          # Compute stochastic growth rate
[1] 0.02552


# Plot graph with the estimated population growth rates and the population
size
par(mfrow = c(1,2), las = 1)
plot(r, type = "l", lwd = 1.5, ylab = "Annual population growth rate", xlab =
      "Time")
abline(h = mean(r))
plot(N[1,], type = "l", lwd = 1.5, ylab = "Population size", xlab = "Time")
lines(N[2,], lwd = 1.5, col = "red")
legend("topleft", lwd = c(1.5, 1.5), col = c("black", "red"), legend = c("1-
      year old", ">1-year old"), bty = "n")
```

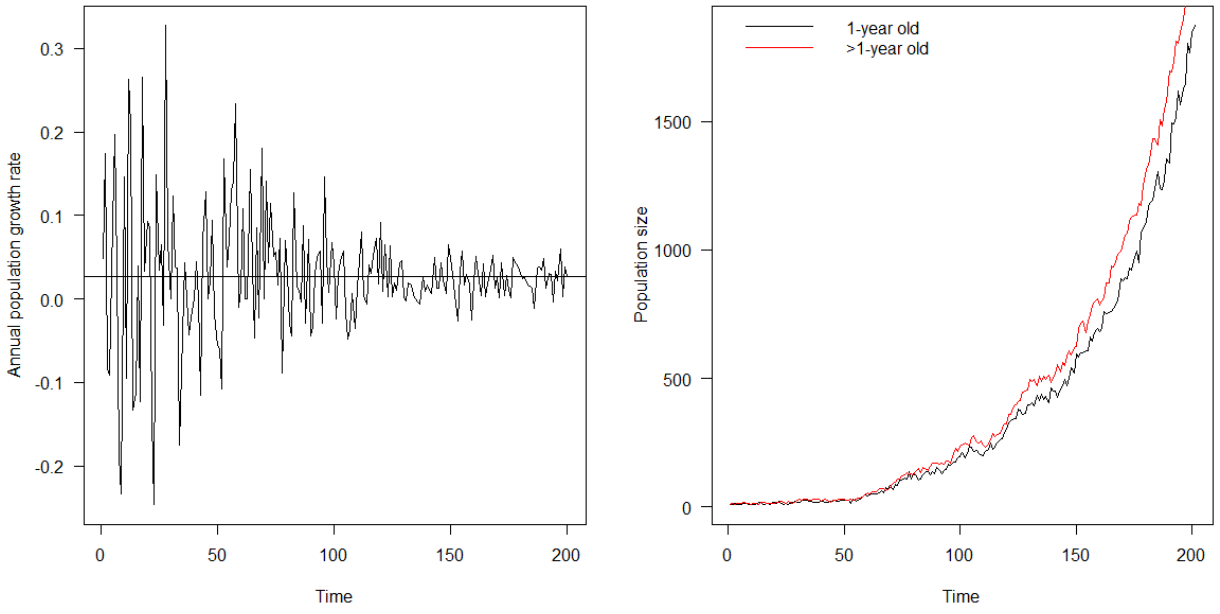


Figure 3.11 Trajectory of estimated annual population growth rate (with average superimposed as a horizontal line) and the estimated, stage-specific population size in a woodchat shrike population that is subject to demographic stochasticity. The demographic parameters are constant over time and not subject to estimation error. The starting population size was 20 females, with half each in the two age classes.

The trajectory of the annual population growth rates is shown graphically in Fig. 3.11. We notice that the growth rate is much more variable at the beginning than later during the simulation. This is exactly due to demographic stochasticity – as the population is increasing in size, its growth rate becomes less variable, and the less relevant demographic stochasticity becomes for population modelling.

Although we have projected the population for 200 year, the current result (for any given year) is a single realization of the stochastic process represented by the model. If we want to obtain a measure of the mean and the precision of the population growth rate, we need to repeat the projection many times, i.e., to perform a Monte Carlo simulation. When we do that we can also evaluate how many times the population went extinct within a specified time frame. The frequency of these extinction events is interpreted as the extinction probability. Here we repeat the projection 10000 times.

```
# Define mean of the demographic parameters
mean.sj <- 0.3
mean.sa <- 0.55
mean.f1 <- 1.3
mean.fa <- 1.8

# Define the number of years with predictions and the Monte Carlo setting
T <- 200
nsim <- 10000

# Define population matrix and initial stage-specific population sizes
N <- array(NA, dim = c(2, T+1, nsim))
N[,1,] <- c(10, 10)
r <- matrix(NA, nrow = T, ncol = nsim)
alive <- matrix(NA, nrow = T, ncol = nsim)
mean.r <- numeric()

# Project population
for (s in 1:nsim){
   if(s %% 1000 == 0) {cat(paste("*** Simrep", s, "***\n")) }  # Counter
   for (t in 1:T){
      N[1,t+1,s] <- rpois(1, mean.sj * mean.f1 * N[1,t,s] + mean.sj * mean.fa
* N[2,t,s])
      N[2,t+1,s] <- rbinom(1, sum(N[,t,s]), mean.sa)
      if (sum(N[,t+1,s]) == 0) break
      else {
         r[t,s] <- log(sum(N[,t+1,s])) - log(sum(N[,t,s]))
         alive[t,s] <- t
         } # else
      } # t
   mean.r[s] <- mean(r[min(alive[,s], na.rm = TRUE):max(alive[,s], na.rm =
TRUE),s])
   } # s
```

The annual population growth rates of all simulated populations is shown in Fig. 3.12 (a). The variability of the growth rates is larger at the beginning than later. This pattern occurs because most populations are increasing and the impact of demographic stochasticity on these population declines, which is a typical feature of demographic staochasticity.

The mean and the standard deviation of the stochastic population growth are computed as follows:

```
# Mean and SD of the population growth rate
mean(mean.r)
```

```
[1] -0.00919845
sd(mean.r)
[1] 0.05762397
```

We notice that the mean across all 10,000 simulated populations is markedly different from the mean of a single realisation of the same process that we had before. The reason for this discrepancy is that the former includes also the mean growth rates of populations that went extinct. The frequency distribution of the growth rates is clearly bimodal (Fig. 3.12 b) and the aritmetic mean is not a good description of this distribution. Often we want to get a population growth rate only for those populations that survived over the defined time threshold, which is computed as:

```
# Mean and SD of the population growth rate for populations that did not went extinct
not.extinct <- which(!is.na(alive[T,]))
mean(mean.r[not.extinct])
[1] 0.01976801
sd(mean.r[not.extinct])
[1] 0.005960932
```

The mean population growth rate of the non-extinct populations is now closer to that of the single realization and also to the asymptotic population growth rate (recall, it is log(1.0208) = 0.0206). Also note that the populations that did not go extinct have a much smaller variability than those that did go extinct. The frequency distribution of the population growth rates of the non-extinct populations is unimodal, but slightly skewed (Fig. 3.12 c).

The simulations also allow us to evaluate the extinction probability:

```
# Extinction probability (after T years)
sum(is.na(alive[T,])) / nsim
[1]  0.291
```

It is possible to calculate also other statistics related to the extinction probability, such as the time to extinction for the extinct populations, or the sensitivity of the extinction probability to changes in demographic rates or initial population size. Here we defined extinction as the event that a population dropped exactly to zero, but in practice we often choose an extinction threshold that is larger than zero. Quasi-extinction is then defined as the event that a population size drops below that threshold. We will discuss these issues more deeply in chapter 10.

```
# Plot graph with population growth rates (Fig. 3.12)
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
par(las = 1, cex = 1.1)
plot(r[,1], type = "l", lwd = 0.5, ylab = "Annual population growth rate",
xlab = "Time", ylim = range(r[which(!is.na(alive))]), col = "lightgrey")
for (s in 2:nsim){
   lines(r[!is.na(alive[,s]),s], lwd = 0.5, col = "lightgrey")
   }
lines(apply(r, 1, mean, na.rm = TRUE), lwd = 1.5)
legend("bottomright", lwd = c(1,1), col = c("lightgrey", "black"), legend =
      c("Individual", "Mean"), bty = "n")
text(x = 0, y = 1.3, "a")
a <- hist(mean.r, nclass = 25, col = "dodgerblue", main = "", xlab =
      "Population growth rate")
text(x = a$mids[1], y = max(a$counts)*0.95, "b")
box()
```

```
a <- hist(mean.r[not.extinct], nclass = 25, col = "dodgerblue", main = "",
      xlab = "Population growth rate")
text(x = a$mids[1], y = max(a$counts)*0.95, "c")
box()
```

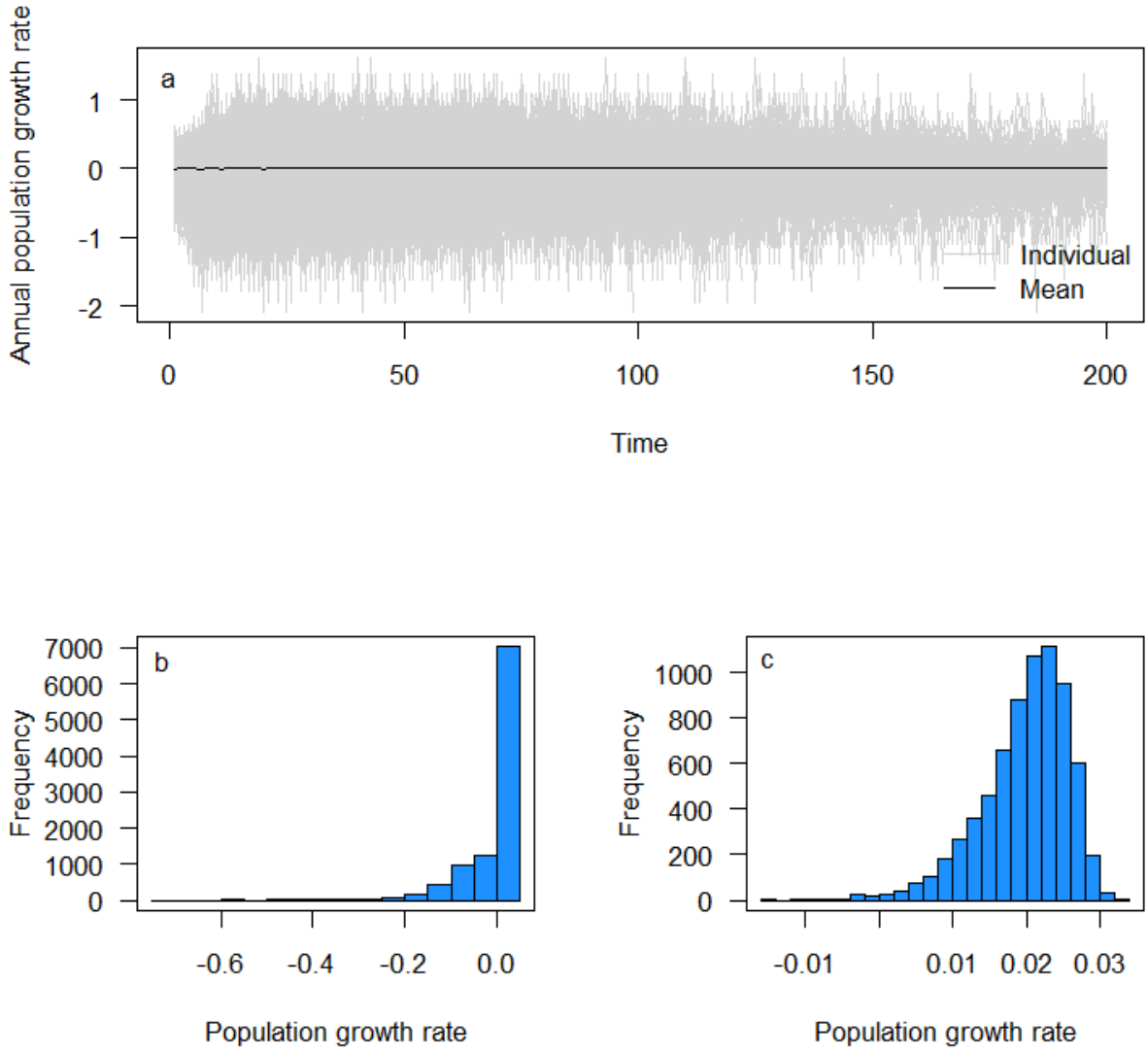


Figure 3.12 Population growth rates of a shrike population that includes demographic stochasticity. a) Realized annual population growth rates of all 10,000 populations (grey) along with the mean (black) across all populations. b) Frequency distribution of the mean growth rates of all populations. c) Frequency distribution of the mean growth rates of the non-extinct populations only.

Other life-history summaries such as the stage-distribution or growth rate sensitivities can also be computed by adapting the code in an analogous way as we did in chapter 3.3.3.

3.3.5. Analysis of a projection matrix model with different sources of stochasticity and parameter uncertainty

A further development of the population model is of course to include both types of stochasticity (environmental and demographic) and parameter uncertainty. Conceptually this is not a big deal, but it does require more effort at book-keeping for all modelled quantities. To illustrate how such a model works, we expand on the previous examples. We assume that the means of survival and productivity are subject to estimation error and that in addition we have an estimate of the temporal variability of the true values of these rates (i.e., igoring the estimation error for the sake of simplification). The initial population size is again 20 females (see exercise XY for accommodating estimation error in the initial population sizes as well). To estimate the population growth rate and life-history summaries we again need to project many populations over a large number of years and summarize the mean behaviour of them. Here is a pseudo-code representation of the algoritm:

1. Draw a value for each demographic rate from a statistical distribution that reflects the estimation uncertainty for the means of the demographic rates.
2. Draw annual demographic rates from appropriate distributions reflecting temporal variability using the means obtained in 1 and the temporal variability (which is assumed known).
3. Loop over time, i.e., let the population evolve over time: determine the number of individuals in each year using appropriate statistical distributions that reflect the demographic processes in each year (and therefore account for demographic stochasticity). Use the values of the annual rates there were obtained in step 2.
4. Repeat steps 1-3 a large number of times (for many populations).

Here are the parameter values that we will use.

```
# Define mean, measurement error and temporal variability of the demographic
parameters
mean.sj <- 0.3      # Mean value of juv. survival
sd.sj.e <- 0.005    # Uncertainty of mean juv. survival expressed as SD on
natural scale
sd.sj.t <- 0.25     # Temporal variability of juv. survival expressed as SD on
logit scale
mean.sa <- 0.55     # Mean value of ad. survival
sd.sa.e <- 0.005    # Uncertainty of mean ad. survival expressed as SD on
natural scale
sd.sa.t <- 0.07     # Temporal variability of ad. survival expressed as SD on
logit scale
mean.f1 <- 1.3      # Mean value of fecundity of 1y females
sd.f1.e <- 0.05     # Uncertainty of mean fecundity expressed as SD on natural
scale
sd.f1.t <- 0.3      # Temporal variability of fecundity expressed as SD on
natural scale
mean.fa <- 1.8      # Mean value of fecundity of adult females
sd.fa.e <- 0.03     # Uncertainty of mean fecundity expressed as SD on natural
scale
sd.fa.t <- 0.3      # Temporal variability of fecundity expressed as SD on
natural scale

# Define the number of years with predictions and the Monte Carlo setting
T <- 200            # Number of years (projection time frame)
nsim <- 1000        # Number of replicate populations simulated
```

```
# Define population matrix and initial stage-specific population sizes
N <- array(NA, dim = c(2, T+1, nsim))
N[,1,] <- c(10, 10)
r <- matrix(NA, nrow = T, ncol = nsim)
alive <- matrix(NA, nrow = T, ncol = nsim)
mean.r <- numeric()

# Project population
for (s in 1:nsim){            # Loop over replicate populations
   if(s %% 100 == 0) {cat(paste("*** Simrep", s, "***\n")) }  # Counter
   # Generate a mean of the demographic rates (subject to measurement error)
   msj <- rbeta(1, beta.params(mean.sj, sd.sj.e)$alpha, beta.params(mean.sj,
sd.sj.e)$beta)
   msa <- rbeta(1, beta.params(mean.sa, sd.sa.e)$alpha, beta.params(mean.sa,
sd.sa.e)$beta)
   mf1 <- rnorm(1, mean.f1, sd.f1.e)
   mfa <- rnorm(1, mean.fa, sd.fa.e)

   # Generate annual demographic rates (subject to temporal variability)
   sj <- plogis(rnorm(T, qlogis(msj), sd.sj.t))
   sa <- plogis(rnorm(T, qlogis(msa), sd.sa.t))
   f1 <- rnorm(T, mf1, sd.f1.t)
   fa <- rnorm(T, mfa, sd.fa.t)

   # Project population (include demographic stochasticity)
   for (t in 1:T){         # Loop over years
      N[1,t+1,s] <- rpois(1, sj[t] * (f1[t] * N[1,t,s] + fa[t] * N[2,t,s]))
      N[2,t+1,s] <- rbinom(1, sum(N[,t,s]), sa[t])
      if (sum(N[,t+1,s]) == 0) break
      else {
         r[t,s] <- log(sum(N[,t+1,s])) - log(sum(N[,t,s]))
         alive[t,s] <- t
         } # else
      } # t
   mean.r[s] <- mean(r[min(alive[,s], na.rm = TRUE):max(alive[,s], na.rm =
TRUE),s])
   } # s
```

Note that it can happen that there is an error in the computation. This is due to the function `rbinom` in R, which crashed and produces an NA when a binomial total gets larger than about 2^{31}. A workaround is to exlude demographic stochasticity after the population has reached a very large size, because then it simply does not matter any more. Here, if you get this problem, we suggest you simply execute the code again until it runs without problems. The mean population growth rate is calculated as

```
mean(mean.r)
-0.01315699
sd(mean.r)
[1] 0.0658146
```

The distribution of the mean stochastic population growth rates is strongly skewed (Fig. 3.13 b) and the mean may not be a good descriptor of this distribution. Therefore we may want to compute that mean population growth rate of the non-extinct populations, whose frequency distribution is unmodal and nearly symmetrical (Fig. 3.13. c):

```
not.extinct <- which(!is.na(alive[T,]))
mean(mean.r[not.extinct])
[1] 0.02437027
```

```
sd(mean.r[not.extinct])
[1] 0.01197552
```

The mean stochastic population growth rate of the non-extinct populations is now larger than when stochasticity is ignored. This does not fit with our expectations, but has to do with the fact that we have excluded the extinct populations. We can customize the computation of the mean stochastic population growth as we think it is useful for our purposes. The extinction probability becomes now relatively important.

```
# Extinction probability (after T years)
sum(is.na(alive[T,])) / nsim
[1]  0.362

# Plot graph with population growth rates (Fig. 3.10)
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
par(las = 1, cex = 1.1)
plot(r[,1], type = "l", lwd = 0.5, ylab = "Annual population growth rate",
xlab = "Time", ylim = range(r[which(!is.na(alive))]), col = "lightgrey")
for (s in 2:nsim){
   lines(r[!is.na(alive[,s]),s], lwd = 0.5, col = "lightgrey")
   }
lines(apply(r, 1, mean, na.rm = TRUE), lwd = 1.5)
legend("bottomright", lwd = c(1,1), col = c("lightgrey", "black"), legend =
      c("Individual", "Mean"), bty = "n")
text(x = 0, y = 1.3, "a")
a <- hist(mean.r, nclass = 25, col = "dodgerblue", main = "", xlab =
      "Population growth rate")
text(x = a$mids[1], y = max(a$counts)*0.95, "b")
box()
a <- hist(mean.r[not.extinct], nclass = 25, col = "dodgerblue", main = "",
      xlab = "Population growth rate")
text(x = a$mids[1], y = max(a$counts)*0.95, "c")
box()
```

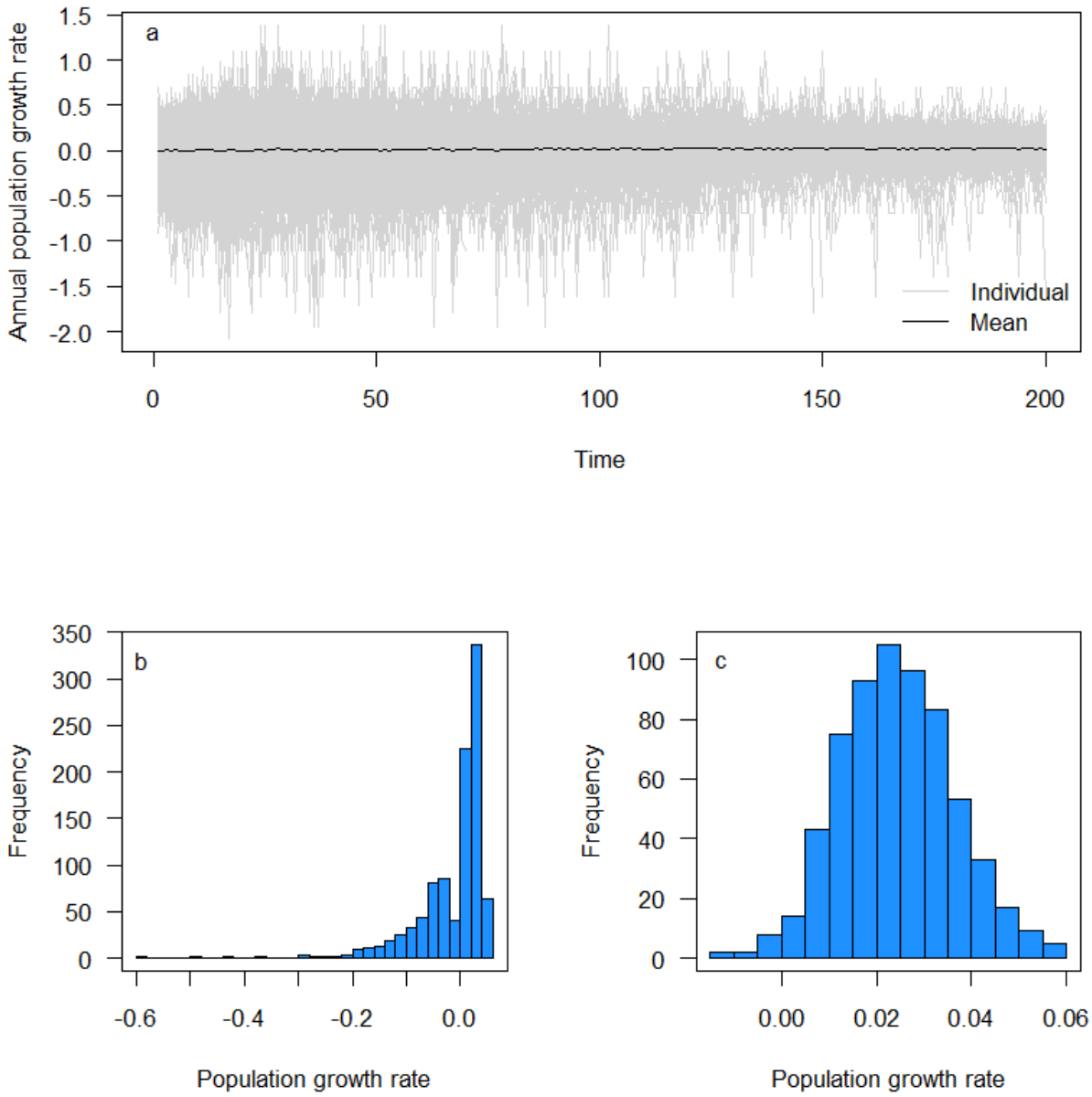


Figure 3.13 Population growth rates in a model that includes environmental and demographic stochasticity, as well as estimation error (parameter uncertainty). a) Realized annual population growth rates of all 1000 populations (grey) along with the mean (black) across the populations. b) Frequency distribution of the mean growth rates of all populations. c) Frequency distribution of the mean growth rates of non-extinct populations.

3.3.6. Population models with density-dependence and demographic stochasticity

All the population models that we have so far presented in this chapter assumed that population growth is independent of the size of the population, i.e., that population growth was exponential. However, natural populations can never grow without limit over a longer time period because resources such as food supply, nest sites or mates always become limited at some point. Therefore, populations in nature always grow in a density-dependent manner, i.e.

as population size increases the population growth rate must decrease (Newton 1998). If a population is regulated by density, the population growth rate λ fluctuates around 1 (or 0 if expressed as *r*).

Different mechanisms can result in a density-dependent response of population growth, yet the basic expectation is always that at least one demographic rate has to decline with increasing population size. Put in a different way, there needs to be a functional relationship (a negative feedback) between population size and such a demographic rate. The population model may then be written in matrix notation as follows: $\boldsymbol{N}_{t+1} = \boldsymbol{A}_N \times \boldsymbol{N}_t$, thus the elements of the transition matrix, and hence the demographic rates, now depend on the population size $\boldsymbol{N}$. The resulting model is nonlinear and the population growth rate and other life history summaries can no longer be written in terms of eigenvalues and eigenvectors. The easiest way to study a population model that includes density-dependence is once more simulation.

There are various different mathemathical forms with which the relationship between population size and a demographic rate can be described (e.g. Ricker, Gompertz, Beverton-Holt' reference here). This is not surprising, since there are literally an infinity of possible curves relating a demographic rate and population size with the simple desired characeristics, that they be monotonic and negative. Here, for simplicitly, we assume a linear relationship.

The typical behavior of most populations that are regulated by density is that their sizes reach an equilibrium density (typically referred to as the *carrying capacity*) which do not change anymore, provided of course that the populations are not perturbated further. Some populations, under density-dependence, may also show cycles or even chaotic behavior (Caswell 2001). Of course, at equilibrium the population growth rate is $\lambda = 1$ (or $r = 0$), and thus there is not much interest in estimating this quantity when there is density dependence. However, there might be an interest in estimating carrying-capacity or the time needed until the population reaches the carrying-capacity. Since the population size obviously matters when density-dependence is modeled, we typically include demographic stochasticity. This requires Monte Carlo simulations to assess the behavior of the population.

We here assume that the available territories in our woodchat shrike population differ in quality, such that reproductive success also differs between individuals. High-quality territories allow raising a large number of fledglings and are occupied in most years, regardless of the actual population size. In contrast, low-quality territories typically produce fewer nestings and they typically are only occupied when population size is high and all higher quality territories are already occupied. The resulting (population specific) annual fledging productivity will then decline with increasing population size. This population-regulation mechanism has been called site-dependent population regulation or buffer effect and has been found repeatedly in territorial species (Rodenhouse et al. 1997, Gill et al. 2001). In our simulation, we will not describe each individual territory, but rather add a functional relationship between population size and fledging production; since this is the net result of heterogeneity in territory quality and nestling output. We also assume that 1-year females are weaker competitors than older females and thus induce stronger density-dependence for f_1 than for f_a. Survival probabilities are assumed to be unaffected by population size and to be constant over time. Thus, in this simulation we assume that density-dependence only acts on fecundity, not on survival. We initialize our population with 20 females, we predict the population over 200 years and repeat this for 1000 replicate populations. The R code for these simulations should pose no major difficulty. Perhaps we need to note that fecundity in each year can only be calculated when the population size known, thus its computation needs to be included in the loop over time.

```
# Define means of the demographic rates and strength of density-dependence
mean.sj <- 0.3
```

```
mean.sa <- 0.55
f1.int <- 2.3           # Fecundity of 1y females when population size is 0
f1.beta <- -0.02        # Strength of density-dependence on fecundity of 1y
                        # This is simply the slope of the regression of f1 on N
fa.int <- 2.3           # Fecundity of adult females when population size is 0
fa.beta <- -0.01        # Strength of density-dependence on fecundity of adults

# Define the number of years with predictions
T <- 200
nsim <- 1000

# Define population matrix and initial stage-specific population sizes
N <- array(NA, dim = c(2, T+1, nsim))
N[,1,] <- c(10, 10)
r <- f1 <- fa <- matrix(NA, nrow = T, ncol = nsim)

# Project population
for (s in 1:nsim){
   if(s %% 100 == 0) {cat(paste("*** Simrep", s, "***\n")) }  # Counter
   for (t in 1:T){
      f1[t,s] <- f1.int + f1.beta * sum(N[,t,s])
      fa[t,s] <- fa.int + fa.beta * sum(N[,t,s])
      N[1,t+1,s] <- rpois(1, mean.sj * (f1[t,s] * N[1,t,s] + fa[t,s] *
N[2,t,s]))
      N[2,t+1,s] <- rbinom(1, sum(N[,t,s]), mean.sa)
      } # t
   } # s


# Fig. 3.14
par(mfrow = c(3,1), mar = c(3,5,3,1), las = 1)
plot(colSums(N[,,1]), type = "l", col = "lightgrey", ylim = c(0, 120), ylab =
"Population size", xlab = "")
for (s in 2:nsim){
   lines(colSums(N[,,s]), col = "lightgrey")
   }
Nall <- apply(N, c(2,3), sum)
lines(apply(Nall, 1, mean), col = "black", lwd = 1.5)
legend("topright", legend = c("Individual", "Mean"), lty = c(1, 1), col =
c("lightgrey", "black"), bty = "n")
text(y = 120, x = 1, "a")

par(mar = c(5,5,1,1))
f1[f1==f1.int] <- NA
plot(f1[,1], type = "l", col = "lightgrey", ylim = c(0.3, 2.3), ylab =
"Fecundity 1y females", xlab = "Time")
for (s in 2:nsim){
   lines(f1[,s], col = "lightgrey")
   }
lines(apply(f1, 1, mean, na.rm = T), col = "black", lwd = 1.5)
text(y = 2.3, x = 1, "b")

par(mar = c(5,5,1,1))
fa[fa==fa.int] <- NA
plot(fa[,1], type = "l", col = "lightgrey", ylim = c(0.3, 2.3), ylab =
"Fecundity adult females", xlab = "Time")
for (s in 2:nsim){
   lines(fa[,s], col = "lightgrey")
   }
lines(apply(fa, 1, mean, na.rm = T), col = "black", lwd = 1.5)
text(y = 2.3, x = 1, "c")
```

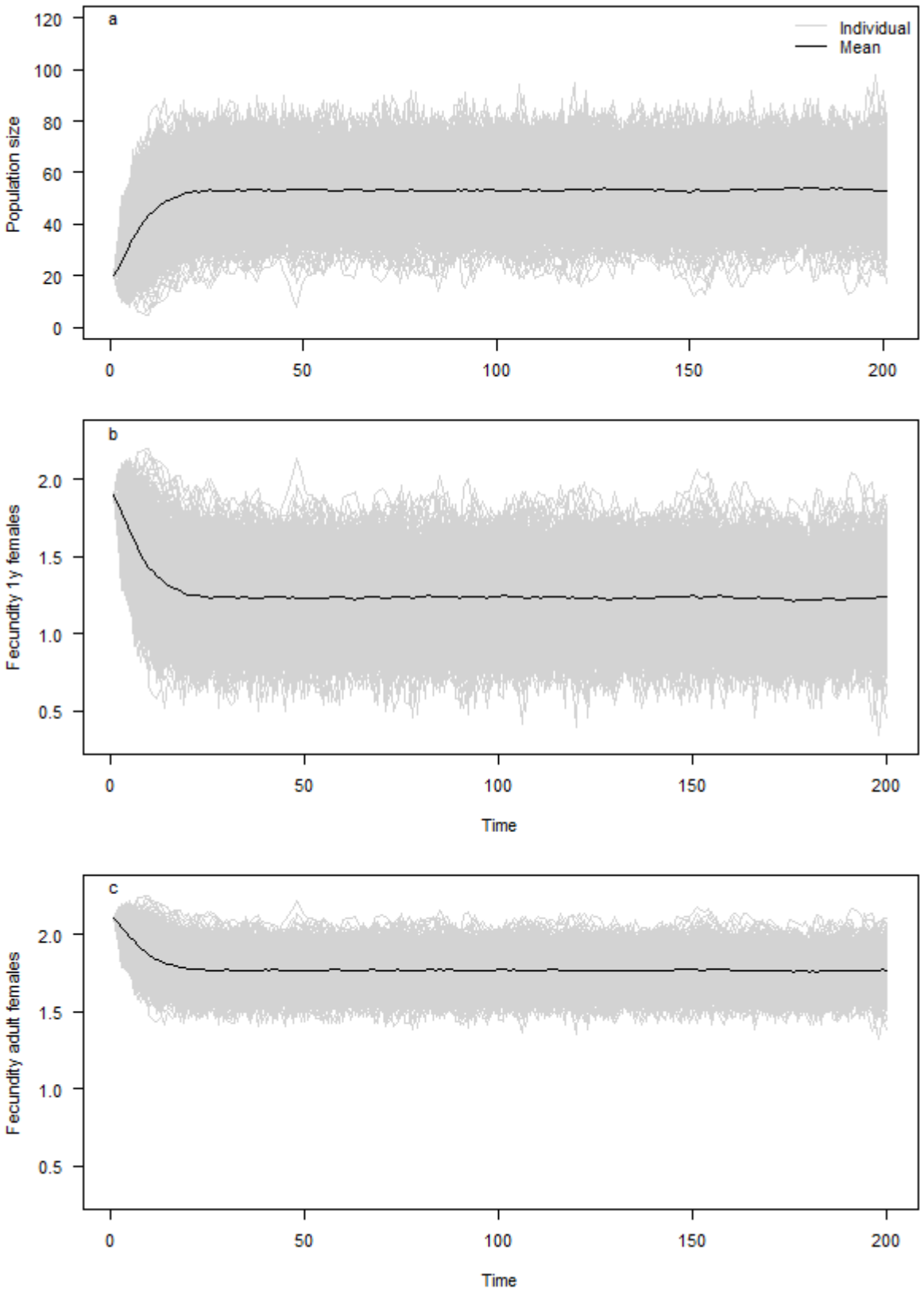


Figure 3.14 Trajectories of (a) population size and fecundity of 1y old (b) and adult (c) females for our experimental population of woodchat shrikes which is subject to density-dependent fecundity and demographic stochasticity (survival is density-independent). Plot shows the trajectories for 1000 simulated populations as well as their mean.

From Fig. 3.14 we see the expected patterns of population size and fecundity: population size approaches the carrying capacity and then remains more or less stable, with fluctuations that are due to demographic stochasticity in this moderately small population. In contrast, fecundity is declining with increasing population size and then stabilizes as the carrying capacity is attained. As an estimate of carrying capacity we can use the mean of the population sizes after 200 time intervals:

```
mean(Nall[T+1,])
[1] 52.92
```

The average fecundities at carrying-capacity are

```
mean(f1[T,], na.rm=T)
[1] 1.23696
mean(fa[T,], na.rm=T)
[1] 1.76848
```

In addition, we also note that some simulated populations went extinct (Fig. 3.14 a). The extinction probability (evaluated from the 1000 replicate populations) is small though:

```
mean(Nall[201,] == 0)
[1] 0
```

It is straightforward to include also environmental stochasticity, estimation errors of the parameters, density-dependence in other demographic rates or different functional forms of density-dependence. The sensitivity of the carrying capacity to changes in the demographic rates can also be calculated using Monte Carlo simulations in the same way as shown before.

3.4. Analysis of matrix projection models with Markov Chain Monte Carlo (MCMC) software

We have seen how fundamentally important simulation is for the analyses of all but the simplest population models. Interestingly, simulation techniques are also widely used in modern statistical modeling for parameter estimation, especially the powerful Bayesian technique known as Markov chain Monte Carlo (MCMC; add some references here). Here, we make the transition to using such MCMC software for doing all of the classical analyses of matrix projection models that we have described in the previous section. We will see that this has some advantages in itself, because of course any analysis done with MCMC is inherently simulation-based (now, that's a truism ....). But, much more importantly, MCMC is our method of choice for parameter estimation in statistical models; for many examples of the power of Bayesian population analyses, see for instance Kéry & Schaub (2012). So the main reason for why we now transition to doing simple analyses of matrix projection models with MCMC software (specifically, with the JAGS variant of BUGS) is that this will then later allow us the seamless integration of matrix projection modeling and parameter estimation, and this integration is one of the key defining features of integrated population models (IPMs).

To illustrate the use of MCMC methods implemended in BUGS we start as before with a model that does neither include stochasticity nor measurement errors of the demographic rates, and then relax these assumptions. The goal is again the computation of the population growth rate and of various life history summaries. The principle that is applied when matrix projection models are analysed with BUGS is that the population is projected over a couple of years, as is necessary in the classical approach when environmental or demographic stochasticity is included. From the projected population trajectories, the stage specific

population sizes and many quantities of interest can be derived. A key advantage of using the MCMC machinery for computing these quantities is that the uncertainty of the population growth rate or of life history summaries are automatically and correctly provided when the demographic parameters contain estimation errors and/or are subject to stochasticity. Clearly, this approach is a bit clumsy for simple cases, but the nice feature is that this general approach is valid for any kind of complexity we want to model. Moreover, the models that we develop here can be extended to integrated population models in a very natural way and it makes the link between IPM and matrix projection models very evident.

3.4.1. Analysis of a matrix projection model without stochasticity and parameter uncertainty

We again assume that the demographic rates of our woodchat shrikes are constant over time and that they are known perfectly, i.e. without parameter uncertainty. Unfortunately, hardly any matrix operations are available in BUGS or JAGS (though many more in NIMBLE; have to check that later). Hence, we will not use matrix multiplication for projecting the population, but the stage-specific equations. As we have seen already in chapters 3.2.3 the number of 1-year old individuals in year t+1 ($N_{1,t+1}$), as a function of the modelled quantities in year t, is

$$N_{1,t+1} = N_{1,t} f_1 s_j + N_{2,t} f_a s_j .$$

Likewise, the number of individuals at least 2 years old in year t+1 ($N_{2,t+1}$) is

$$N_{2,t+1} = N_{1,t} s_a + N_{2,t} s_a = \left(N_{1,t} + N_{2,t}\right) s_a .$$

To project the population size, we must know (or make a decision about) the stage-specific population size at time t = 1. As this is unknown, we start with arbitrary values. From theory we know (and we have demonstrated in section 3.3.1) that if the demographic rates are constant, the population will grow at a constant rate once it has reached a stationarity, i.e., stable stage distribution, and that under conditions that must be assumed to be very mild, convergence to stationarity is virtually certain (Caswell 2001). That is, if the stable stage distribution is known we only need to project the population for a single time step to obtain an estimate of the asymptotic growth rate. But since we don't know the stable stage distribution, we must start with an arbitrary population vector and project the population for a number of time steps until the Markov chain has reached stationarity. We have seen before that this can happen in very short time in simple models such as ours. If the stable stage distribution is obtained after T projections, an estimate of the asymptotic population growth rate is

$$\lambda = \frac{N_{1,T+1} + N_{2,T+1}}{N_{1,T} + N_{2,T}} = \frac{N_{1,T+1}}{N_{1,T}} = \frac{N_{2,T+1}}{N_{2,T}} .$$

In practice we do not know T, but we can simply project the population long enough and see when the growth rate estimates stabilize at some constant value.

Other life history summaries can be calculated as explained in chapters XY. In the following BUGS model file the population is projected for T years and the asymptotic population growth rate is calculated.

The data consist of the demographic parameters and the desired number of years of population projection (T). Here, T can be very small (T = 5), but when we model species with

longer generation times or if our model contains stochasticity, longer runs are needed to obtain estimates with reasonable precision.

```
# Bundle data
sj <- 0.3
sa <- 0.55
f1 <- 1.3
fa <- 1.8

bugs.data <- list(sj = sj, sa = sa, f1 = f1, fa = fa, T = 5)

# Write BUGS model file
cat(file = "m0.jags", "
model {
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1
dummy ~ dunif(0, 1) # Dummy stochastic node needed (see text for more
explanation)

# Loop over time
for (t in 1:T){
   # Population projection
   N[1,t+1] <- sj * (f1 * N[1,t] + fa * N[2,t])
   N[2,t+1] <- sa * (N[1,t]+ N[2,t])

   # Calculation of population quantities
   # Annual (realized) population growth rate
   ann.growth.rate[t] <- (N[1,t+1]+ N[2,t+1]) / (N[1,t]+ N[2,t])
   }
lambda <- ann.growth.rate[T] # gr in final interval is our estimate of lambda
}
")


# Initial values
inits <- function(){list(dummy = 0.5)}

# Parameters monitored
parameters <- c("ann.growth.rate", "lambda", "N")

# MCMC settings
ni <- 2; nt <- 1; nb <- 0; nc <- 1

# Call JAGS from R and summarize posteriors
m0 <- jags(bugs.data, inits, parameters, "m0.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m0, 3)

JAGS output for model 'm0.jags', generated by jagsUI.
Estimates based on 1 chains of 2 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 2 total samples from the joint posterior.
MCMC ran for 0.006 minutes at time 2016-08-15 16:08:39.

                    mean sd  2.5%   50% 97.5% overlap0 f
ann.growth.rate[1] 1.015  0 1.015 1.015 1.015    FALSE 1
ann.growth.rate[2] 1.021  0 1.021 1.021 1.021    FALSE 1
ann.growth.rate[3] 1.021  0 1.021 1.021 1.021    FALSE 1
ann.growth.rate[4] 1.021  0 1.021 1.021 1.021    FALSE 1
ann.growth.rate[5] 1.021  0 1.021 1.021 1.021    FALSE 1
```

```
lambda              1.021   0 1.021 1.021 1.021    FALSE 1
N[1,1]              1.000   0 1.000 1.000 1.000    FALSE 1
N[2,1]              1.000   0 1.000 1.000 1.000    FALSE 1
N[1,2]              0.930   0 0.930 0.930 0.930    FALSE 1
N[2,2]              1.100   0 1.100 1.100 1.100    FALSE 1
N[1,3]              0.957   0 0.957 0.957 0.957    FALSE 1
N[2,3]              1.117   0 1.117 1.117 1.117    FALSE 1
N[1,4]              0.976   0 0.976 0.976 0.976    FALSE 1
N[2,4]              1.140   0 1.140 1.140 1.140    FALSE 1
N[1,5]              0.996   0 0.996 0.996 0.996    FALSE 1
N[2,5]              1.164   0 1.164 1.164 1.164    FALSE 1
N[1,6]              1.017   0 1.017 1.017 1.017    FALSE 1
N[2,6]              1.188   0 1.188 1.188 1.188    FALSE 1
```

All estimates of the annual population growth rates are identical, hence, the Markov chain has reached its stationary distribution almost from the first iteration (projection). This corresponds exactly to our findings from before (see Fig. XY). We use the values from the last time step as our estimates of the asymptotic population growth rate (`lambda`). This value is exactly identical to what we obtained from the dominant eigenvalue of the transition matrix (chapter 3.3.1). All the estimates have a posterior standard deviation (the Bayesian analogue of the standard error) of 0. This makes sense, because the model is deterministic and includes neither stochasticity nor uncertainty of the demographic rates.

We make a few remarks about the code, the function `jags` in the R package `jagsUI` (Kellner 2014) and our MCMC settings. First, in BUGS software we cannot run models that are entirely deterministic, rather, we need at least one stochastic node in the model. So to trick BUGS into doing the calculations of our deterministic model, we simply add one arbitrary stochastic node by drawing a quantity called 'dummy' from a standard uniform distribution. Later on, when we analyse models with some stochastic elements, we will no longer need anymore such a dummy stochastic node.

Second, we turn off calculation of the deviance information criterion (DIC), when jags is called, because the deviance cannot be calculated in this model. This is the `DIC = FALSE` part in the function call.

Third, our results are for a deterministic model, thus we do not need replicates. Yet, the MCMC machinery makes replicates (simulations). We choose the MCMC settings such that only as few simulations as necessary possible are performed. Therefore, we don't do a burn-in, we ask for 1 chain only, and we only iterate the MCMC alrgorithm for 2 steps (with just 1 sample JAGS returns an error).

Finally, we note that this is not yet a Bayesian analysis of the model in the usual sense of a Bayesian analysis. There is no uncertainty about anything in this deterministic model, and we simply use the MCMC machinery to calculate the desired quantities.

We can now extend this code to calculate from the same data other life-history summaries as well. Applying the same logic as for the asymptotic population growth rate, we use the stage distribution in the final year *T* as an estimate of the stable stage distribution. The sensitivity of the population growth rate to changes in juvenile survival may be calculated as stochastic sensitivities (chapter XY), although there is no stochasticity here. However, stochasticic sensitivities are always correct, even in the absence of environmental stochasticity. Finally we also compute the net reproductive rate (R_0) and the generation time (*GT*) based on formulae XY and XY, respectively.

```
# Write BUGS model file
cat(file = "m1.jags", "
```

```
model {
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1
dummy ~ dunif(0, 1)   # Dummy stochastic node to keep JAGS happy

# Loop over time
for (t in 1:T){
   # Population projection
   N[1,t+1] <- sj * (f1 * N[1,t] + fa * N[2,t])
   N[2,t+1] <- sa * (N[1,t]+ N[2,t])

   # Calculation of population quantities
   # Annual (realized) population growth rate
   ann.growth.rate[t] <- (N[1,t+1]+ N[2,t+1]) / (N[1,t]+ N[2,t])

   # Scaled annual stage distributions
   stage.distr[1,t] <- N[1,t+1] / (N[1,t+1]+ N[2,t+1])
   stage.distr[2,t] <- N[2,t+1] / (N[1,t+1]+ N[2,t+1])
   }
lambda <- ann.growth.rate[T]
stable.stage.distr <- stage.distr[,T]

# Sensitivity and elasticity of lambda to changes in sj
delta <- 0.001                     # size of perturbation
N.star[1,1] <- 1
N.star[2,1] <- 1
for (t in 1:T){
   N.star[1,t+1] <- (sj + delta) * (f1 * N.star[1,t] + fa * N.star[2,t])
   N.star[2,t+1] <- sa * (N.star[1,t] + N.star[2,t])
   ann.growth.rate.star[t] <- (N.star[1,t+1]+ N.star[2,t+1]) / (N.star[1,t]+
N.star[2,t])
   }
s.sj <- (ann.growth.rate.star[T] - ann.growth.rate[T]) / delta
e.sj <- s.sj * sj / lambda

# Calculation of net reproductive rate (R0)
for (i in 1:100){
   u[i] <- pow(sa, i)
   }
R0 <- sj * f1  + sj * fa * sum(u[])

# Calculation of generation time (GT)
GT <- log(R0) / log(lambda)
} # end of model
")



# Initial values
inits <- function(){list(dummy = 0.5)}

# Parameters monitored
parameters <- c("lambda", "stable.stage.distr", "s.sj", "e.sj", "R0", "GT")

# MCMC settings
ni <- 2; nt <- 1; nb <- 0; nc <- 1

# Call JAGS from R and summarize posteriors
m1 <- jags(bugs.data, inits, parameters, "m1.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m1, 3)
```

```
JAGS output for model 'm1.jags', generated by jagsUI.
Estimates based on 1 chains of 2 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 2 total samples from the joint posterior.
MCMC ran for 0 minutes at time 2016-08-15 16:18:30.

                        mean sd  2.5%   50% 97.5% overlap0 f
lambda                 1.021  0 1.021 1.021 1.021    FALSE 1
stable.stage.distr[1] 0.461  0 0.461 0.461 0.461    FALSE 1
stable.stage.distr[2] 0.539  0 0.539 0.539 0.539    FALSE 1
s.sj                   1.454  0 1.454 1.454 1.454    FALSE 1
e.sj                   0.427  0 0.427 0.427 0.427    FALSE 1
R0                     1.050  0 1.050 1.050 1.050    FALSE 1
GT                     2.368  0 2.368 2.368 2.368    FALSE 1
```

All quantities computed are identical to those that we obtained using the standard matrix projection methods (chapter 3.3.1). So, remember, for all the calculations performed so far, there is no benefit of using MCMC methods – but there will be one as soon as we include parameter uncertainty into our model, which comes next.

3.4.2. Analysis of a matrix projection model without stochasticity, but with parameter uncertainty

Usually the demographic rates that we fill into the transition matrix of a population model are not known perfectly, but instead they are estimates. For instance, survival probabilities may have be obtained from the fitting of a Cormack-Jolly-Seber (CJS; Pollock et al. 1990; and Chapter XX in this book) model in program MARK (Cooch and White 2015). Thus, all such estimates of demographic rates have an associated estimation error; or, in other words, there is parameter uncertainty in our model, which ought to be accommodated in order for our inferences to be scientifically defensible.

Typically for a demographic rate, we have a point estimate, e.g., the MLE or posterior mean, and an associated measure of uncertainty such as the standard error for MLEs or the posterior standard deviation for a Bayesian estimate. Now, we want to estimate the population growth rate and life history summaries while accounting for that estimation uncertainty. Put in other words, we want to propagate the estimation uncertainty associated with the parameters in the model into derived quantities, such as the population growth rate.

The traditional approach to this is to use simulations (chapter 3.3.2). Using the MCMC machinery in BUGS software we can do such simulations in a very straightforward way, and we don't have to specifically program any simulation algorithm ourselves. We only need to change our model such that the demographic rates are not fixed values anymore, but are characterized by a distribution, which itself represents the uncertainty about the parameter. This distribution is characterized by a mean and a measure of spread that reflects the parameter uncertainty. Using the Bayesian estimation software such as BUGS, we can express the parameter estimation error (i.e. the uncertainty around the value of a parameter) by a statistical distribution for the demographic parameter. Technically, in the BUGS model file this will look exactly like a prior distribution, but it does not act as a prior, because there is still no estimation involved in this model. These "prior" distributions must ensure that no impossible values can be generated. For survival probabilities this means that the drawn values have to be in the interval between 0 and 1, and for productivity only positive values must be produced. Here we use the beta distribution for the survival probabilities and the normal for productivity. Note that the specification of these "prior distributions" is equivalent to the specification of the distributions for the demographic rates from which we generated values in the classical approach in section xx.xx.

For fecundity we can use the Normal distribution to reflect uncertainty, provided that the uncertainty is not too large. Here are means and the associated measures of uncertainty for the four demographic rates in our shrike example.

```
# Define mean and SD of the demographic parameters
mean.sj <- 0.3        # Point estimate of juv. survival
sd.sj.e <- 0.03       # Uncertainty of juv. survival expressed as SD on natural
scale
mean.sa <- 0.55       # Point estimate of ad. survival
sd.sa.e <- 0.015      # Uncertainty of ad. survival expressed as SD on natural
scale
mean.f1 <- 1.3        # Point estimate of fecundity of 1y females
sd.f1.e <- 0.3        # Uncertainty of fecundity expressed as SD on natural
scale
mean.fa <- 1.8        # Point estimate of fecundity of adult females
sd.fa.e <- 0.1        # Uncertainty of fecundity expressed as SD on natural
scale
```

The data bundle includes the shape parameters for the beta distributions and the standard deviations of productivity.

```
# Bundle data
bugs.data <- list(alpha.sj = beta.params(mean.sj, sd.sj.e)$alpha, beta.sj =
beta.params(mean.sj, sd.sj.e)$beta, alpha.sa = beta.params(mean.sa,
sd.sa.e)$alpha, beta.sa = beta.params(mean.sa, sd.sa.e)$beta, mean.f1 =
mean.f1, tau.f1 = 1/sd.f1.e^2, mean.fa = mean.fa, tau.fa = 1/sd.fa.e^2, T =
50)
```

The code to analyse a projection matrix model with uncertainty in the demographic parameters looks almost exactly as before, the only difference being that we specify distributions for the demographic parameters from which one value is randomly drawn at each iteration of the MCMC algorithm. The model now does contain stochastic nodes (the quantites drawn from their "priors"), we no longer need to specify a dummy node.

```
# Write BUGS model file
cat(file = "m2.jags", "
model {
# Priors
sj ~ dbeta(alpha.sj, beta.sj) # These only *look* like priors
sa ~ dbeta(alpha.sa, beta.sa) # ... but they are not
f1 ~ dnorm(mean.f1, tau.f1)   # ... b/c there is no estimation in this model
fa ~ dnorm(mean.fa, tau.fa)   # ... b/c there is no estimation in this model

# Initialize the population size nodes
N[1,1] <- 1
N[2,1] <- 1

# Loop over time
for (t in 1:T){
   # Population model
   N[1,t+1] <- sj * (f1 * N[1,t] + fa * N[2,t])
   N[2,t+1] <- sa * (N[1,t] + N[2,t])

   # Annual (realized) growth rate
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])

   # Scaled stage distributions
```

```
   stage.distr[1,t] <- N[1,t+1] / (N[1,t+1] + N[2,t+1])
   stage.distr[2,t] <- N[2,t+1] / (N[1,t+1] + N[2,t+1])
   }
lambda <- ann.growth.rate[T]
stable.stage.distr <- stage.distr[,T]

# Sensitivity and elasticity of lambda to changes in sj
delta <- 0.001                    # size of perturbation
N.star[1,1] <- 1
N.star[2,1] <- 1
for (t in 1:T){
   N.star[1,t+1] <- (sj + delta) * (f1 * N.star[1,t] + fa * N.star[2,t])
   N.star[2,t+1] <- sa * (N.star[1,t] + N.star[2,t])
   ann.growth.rate.star[t] <- (N.star[1,t+1] + N.star[2,t+1]) / (N.star[1,t] +
N.star[2,t])
   }
s.sj <- (ann.growth.rate.star[T] - ann.growth.rate[T]) / delta
e.sj <- s.sj * sj / lambda

# Calculation of net reproductive rate (R0)
for (i in 1:100){
   u[i] <- pow(sa, i)
   }
R0 <- sj * f1 + sj * fa * sum(u[])

# Calculation of generation time (GT)
GT <- log(R0) / log(lambda)
}
")



# Initial values
inits <- function(){list(sa = runif(1))}

# Parameters monitored
parameters <- c("lambda", "stable.stage.distr", "s.sj", "e.sj", "R0", "GT")

# MCMC settings
ni <- 10000; nt <- 1; nb <- 0; nc <- 1

# Call JAGS from R and summarize posteriors
m2 <- jags(bugs.data, inits, parameters, "m2.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m2, 4)

JAGS output for model 'm2.jags', generated by jagsUI.
Estimates based on 1 chains of 10000 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 10000 total samples from the joint posterior.
MCMC ran for 0.009 minutes at time 2016-08-15 16:27:44.

                        mean     sd   2.5%    50%  97.5% overlap0 f
lambda                1.0218 0.0628 0.9096 1.0184 1.1546    FALSE 1
stable.stage.distr[1] 0.4599 0.0322 0.3982 0.4602 0.5229    FALSE 1
stable.stage.distr[2] 0.5401 0.0322 0.4771 0.5398 0.6018    FALSE 1
s.sj                  1.4630 0.1929 1.1066 1.4550 1.8648    FALSE 1
e.sj                  0.4266 0.0450 0.3422 0.4253 0.5174    FALSE 1
R0                    1.0506 0.1497 0.7767 1.0436 1.3649    FALSE 1
GT                    2.3873 0.1908 2.0546 2.3758 2.7947    FALSE 1
```

Now, our model is no longer strictly deterministic, since there is the stochastic element involved by drawing values from the "prior" distributions, to account for parameter uncertainty. Hence, the column containing the posteriod sd no longer has zero values as in the previous example. These are the correct errors.

To visualize the results we plot the "posterior" distributions of some of the computed quantities (Fig. 3.15). The posterior distributions can be used to make probability statements about the quantities. For example we can calculate the probability that the asymptotic population growth rate is smaller than 1 (i.e. that the population is declining), given the parameter uncertainty in our population model:

```
mean(m2$sims.list$lambda < 1)
[1] 0.3801

# Produce figure 3.15.
par(mfrow = c(2, 2), mar = c(5, 4, 3, 0), las = 1, cex = 1.1)
a <- hist(m2$sims.list$lambda, nclass = 50, col = "dodgerblue", main = "",
xlab = expression(paste("Asymptotic population growth rate (", lambda, ")")),
prob = TRUE)
text(x = a$mids[2], y = max(a$density), "a")
par(mar = c(5, 2, 3, 2))
a <- hist(m2$sims.list$stable.stage[,1], nclass = 50, col = "dodgerblue", main
= "", xlab = "Proportion of 1-year old individuals", prob = TRUE)
text(x = a$mids[2], y = max(a$density), "b")
par(mar = c(5, 4, 3, 0))
a <- hist(m2$sims.list$s.sj, nclass = 50, col = "dodgerblue", main = "", xlab
= expression(paste("Sensitivity of ", lambda, " to variation of juvenile
survival")), prob = TRUE)
text(x = a$mids[2], y = max(a$density), "c")
par(mar = c(5, 2, 3, 2))
a <- hist(m2$sims.list$GT, nclass = 40, col = "dodgerblue", main = "", xlab =
"Generation time", prob = TRUE)
text(x = a$mids[2], y = max(a$density), "d")
```

Since we need to perform simulations here, we increase the size of the number of iterations in the MCMC settings to 10’000, to get 10’000 samples from the posterior distributions. It is still not necessary to include a burn-in or to have multiple chains, because convergence is not an issue in this kind of analysis. After all, it is still not yet a formal Bayesian analysis (because nothing is formally estimated), we still just make use of the MCMC machinery available in JAGS to perform stochastic simulations.

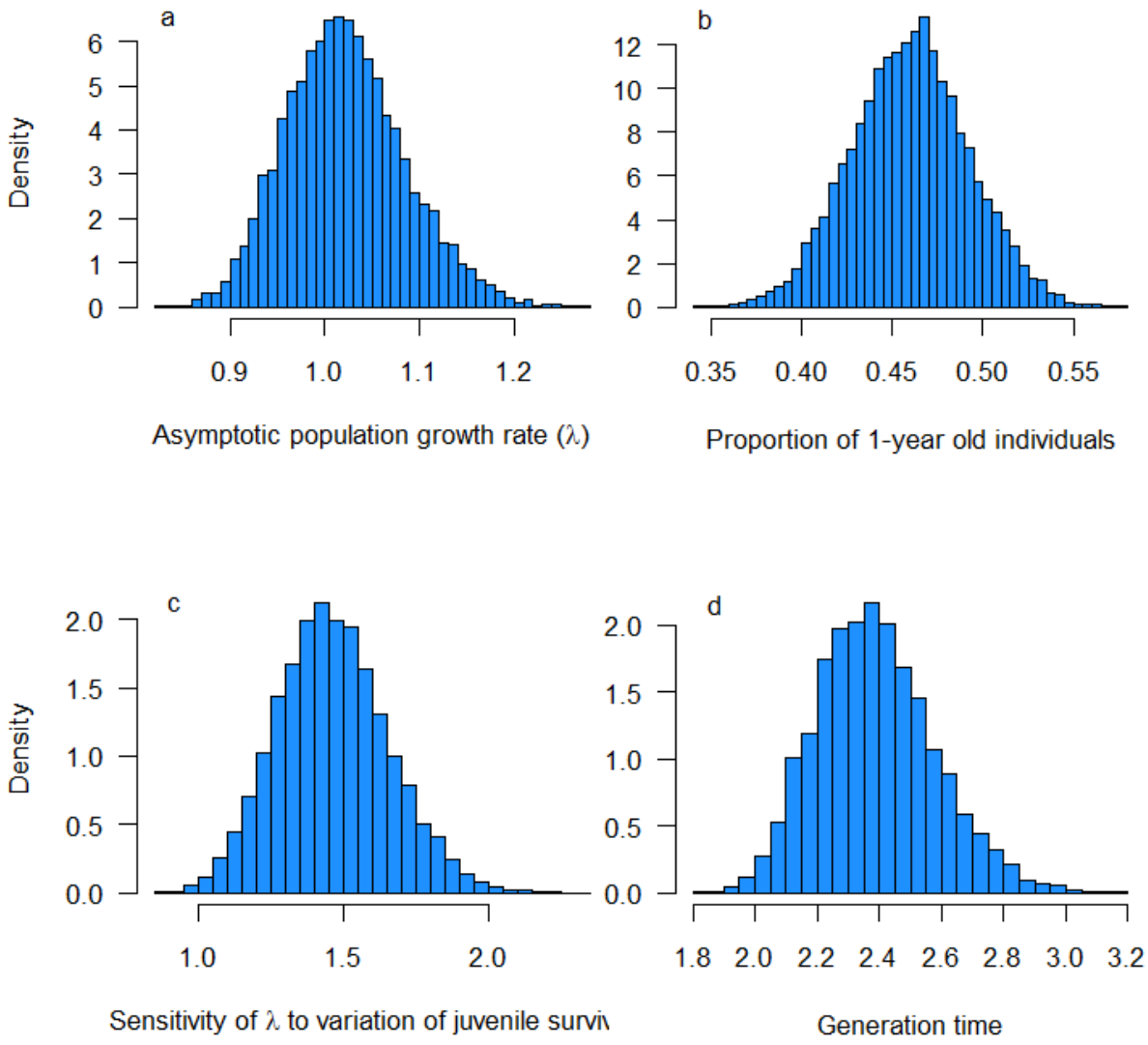


Figure 3.15 Simulated distributions of the asymptotic population growth rate (a), of the proportion of juvenile individuals (b) and of growth rate sensitivity to variation in juvenile survival (c) and of generation time (d) as obtained using MCMC software with 10,000 samples.

The simulated distributions of the target quantities (Fig. 3.15) and the frequency distribution of the target quantities from the simulation that we obtained in chapter 3.3.2 are almost completely identical, as shown by density plots in Fig. 3.16. Slight differences are due to sampling variability; if the number of samples is increased, both distributions will converge to become identical. This means that with both methods, the classical Monte Carlo simulation and with the Bayesian MCMC software, we obtain the same result. This is not a surprise, of course, because mathematically we are doing twice the same (OR similar) thing, but for us it is important to realize this similarity. An advantage of the Bayesian MCMC is that the simulation part is already implemented in JAGS, thus we get the measures of uncertainty without a need to

code up a Monte Carlo simulation. We also note that JAGS is faster than the simulation in R. Both advantages become more important for more complicated situations.

```
par(mfrow = c(2, 2), mar = c(5, 4, 1, 1), las = 1, cex = 1.1)
plot(density(lambda), main = "", xlab = expression(paste("Asymptotic
population growth rate (", lambda, ")")), col = "blue", lwd = 2)
lines(density(m2$sims.list$lambda), col = "red", lwd = 2)
legend("topright", legend = c("classical MC", "Bayesian MCMC"), lwd = c(2,2),
col = c("blue","red"), bty = "n")
par(mar = c(5, 3, 1, 2))
plot(density(stable.stage[,1]), main = "", xlab = "Proportion of 1-year old
individuals", ylab = "", col = "blue", lwd = 2)
lines(density(m2$sims.list$stable.stage[,1]), col = "red", lwd = 2)
par(mar = c(5, 4, 1, 1))
plot(density(sensitivity[,1]), main = "", xlab = expression(paste("Sensitivity
of ", lambda, " to variation of juvenile survival")), col = "blue", lwd = 2)
lines(density(m2$sims.list$s.sj), col = "red", lwd = 2)
par(mar = c(5, 3, 1, 2))
plot(density(GT), main = "", xlab = "Generation time", ylab = "", col =
"blue", lwd = 2)
lines(density(m2$sims.list$GT), col = "red", lwd = 2)
```

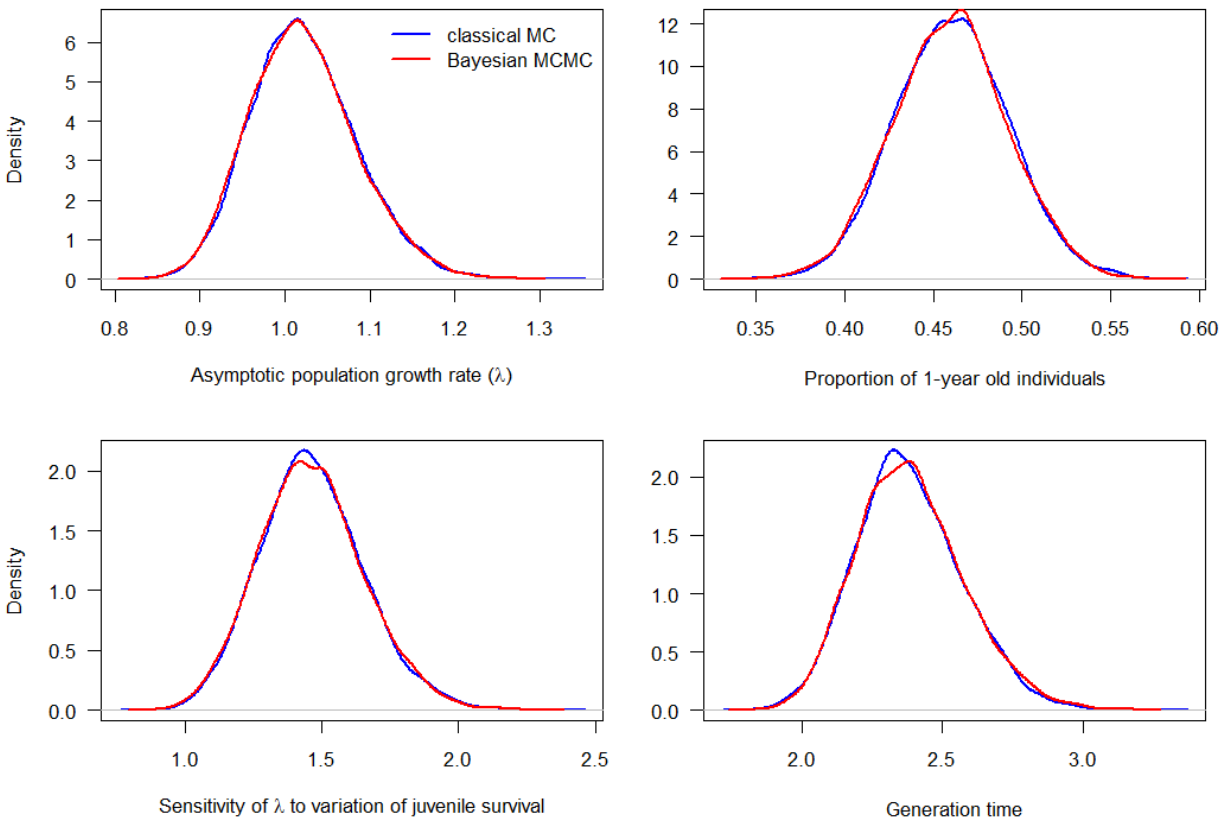


Figure 3.16 Prior distributions of survival and productivity and the resulting posterior distributions for the population growth rate and its sensitivity to changes in juvenile survival (S(sj)).

3.4.3. Analysis of a matrix projection model with environmental stochasticity, but without parameter uncertainty

We next illustrate the case when the demographic rates are subject to environmental stochasticity, i.e., vary over time but are known perfectly. The computation of the stochastic population growth rate and of other life-history summaries are done in the same way as we have already seen for the classical analyses (chapter 3.3.3.): we need to project the population for a long time and from the stage-specific population trajectories we can calculate the quantities of interest.

First we define the mean demographic rates (same values as before) and their temporal variability expressed as standard deviations. We assume that there are no estimation errors in the parameters.

```
# Define mean and temporal variability (SD) of the demographic parameters
mean.sj <- 0.3          # Mean juv. survival on probability scale
sd.sj.t <- 0.25         # Temporal variability of ... on the logit scale
mean.sa <- 0.55         # Mean ad. survival on probability scale
sd.sa.t <- 0.07         # Temporal variability of ... on the logit scale
mean.f1 <- 1.3          # Mean fecundity of 1y females on natural scale
sd.f1.t <- 0.3          # Temporal variability of ... on the natural scale
mean.fa <- 1.8          # Mean fecundity of adult females on natural scale
sd.fa.t <- 0.3          # Temporal variability of ... on the natural scale
```

We will project the population for T = 11,000 years and will discard the first u = 1000 years.

```
# Bundle data
T <- 10000
u <- 1000   # Burn-in to calculate the stochastic population growth rate

bugs.data <- list(mean.sj = mean.sj, mean.sa = mean.sa, mean.f1 = mean.f1,
mean.fa = mean.fa, sd.sj.t = sd.sj.t, sd.sa.t = sd.sa.t, sd.f1.t = sd.f1.t,
sd.fa.t = sd.fa.t, T = T, u = u)
```

The BUGS code differs from the previous examples in important ways. We now need to index by time the demographic rates. To simulate the annual estimates of demographic rates we specify appropriate distributions. That is, we specify an annual value of a demographic rate as a draw from a Normal distribution with the known mean and temporal variability. Since we need to impose the usual range constraints for a probability we draw the annual values of survival on the logit scale and then backtransform, while with fecundity we work directly on the natural scale, both of which as we did previously discussed in the classical analysis. Also as in the classical analysis we compute the stochastic population growth rate and the growth rate sensitivity and elasticity to variation in juvenile survival.

```
# Write BUGS model file
cat(file = "m3.jags", "
model {
# Calculate the mean of the demographic rates on transformed scale
logit.mean.sj <- log(mean.sj / (1-mean.sj))
logit.mean.sa <- log(mean.sa / (1-mean.sa))

# Calculate precision for temporal variability of demographic rates
tau.logit.sj <- pow(sd.sj.t, -2)
tau.logit.sa <- pow(sd.sa.t, -2)
```

```
tau.f1 <- pow(sd.f1.t, -2)
tau.fa <- pow(sd.fa.t, -2)

# Priors
for (t in 1:T){
   sj[t] <- 1/(1+exp(-logit.sj[t]))
   logit.sj[t] ~ dnorm(logit.mean.sj, tau.logit.sj)
   sa[t] <- 1/(1+exp(-logit.sa[t]))
   logit.sa[t] ~ dnorm(logit.mean.sa, tau.logit.sa)
   f1[t] ~ dnorm(mean.f1, tau.f1)
   fa[t] ~ dnorm(mean.fa, tau.fa)
   }

# Model for initial state
N[1,1] <- 1
N[2,1] <- 1

# Loop over time
for (t in 1:T){
   # Population model
   N[1,t+1] <- sj[t] * (f1[t] * N[1,t] + fa[t] * N[2,t])
   N[2,t+1] <- sa[t] * (N[1,t] + N[2,t])

   # Annual growth rate on log scale
   r.annual[t] <- log(N[1,t+1] + N[2,t+1]) - log(N[1,t] + N[2,t])
   }
r <- mean(r.annual[u:T])
lambda <- exp(r)

# Sensitivity and elasticity of lambda to changes in sj
delta <- 0.001                      # size of perturbation
N.star[1,1] <- 1
N.star[2,1] <- 1
for (t in 1:T){
   N.star[1,t+1] <- (sj[t] + delta) * (f1[t] * N.star[1,t] + fa[t] *
N.star[2,t])
   N.star[2,t+1] <- sa[t] * (N.star[1,t] + N.star[2,t])
   r.annual.star[t] <- log(N.star[1,t+1] + N.star[2,t+1]) - log(N.star[1,t] +
N.star[2,t])
   }
r.star <- mean(r.annual.star[u:T])
s.sj <- (exp(r.star)-lambda) / delta
e.sj <- s.sj * mean.sj / lambda
}
")


# Initial values
inits <- function(){list(logit.sj = rnorm(T))}

# Parameters monitored
parameters <- c("r", "lambda", "s.sj", "e.sj")

# MCMC settings
ni <- 2;   nt <- 1;   nb <- 0;   nc <- 1

# Call JAGS from R and summarize simulation results
m3 <- jags(bugs.data, inits, parameters, "m3.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m3, 4)

JAGS output for model 'm3.jags', generated by jagsUI.
Estimates based on 1 chains of 2 iterations,
```

```
burn-in = 0 iterations and thin rate = 1,
yielding 2 total samples from the joint posterior.
MCMC ran for 0.205 minutes at time 2016-08-16 09:10:58.

         mean     sd   2.5%    50%  97.5% overlap0 f
r      0.0194 0.0003 0.0191 0.0194 0.0196    FALSE 1
lambda 1.0195 0.0003 1.0193 1.0195 1.0198    FALSE 1
s.sj   1.4506 0.0027 1.4488 1.4506 1.4524    FALSE 1
e.sj   0.4268 0.0009 0.4262 0.4268 0.4275    FALSE 1
```

We see again that the simulated means of the monitored parameters agree well with the estimates obtained from the classical simulation analysis. Note that standard deviations in the simulation results indicate how well the parameters are estimated which depends, among other things, on the "sample size". Indeed, the precision of these parameters in this model could be made as small as one likes, it is just a matter of the chosen number of years; the standard deviations will converge to zero with increasing number of years. This makes sense, since there is in fact no estimation uncertainty here, all the underlying parameters determining population growth are known without error, and hence the stochastic population growth rate and sensitivities/elasticities do not have any errors associated. This is exactly analogous to the case where we obtain the standard error of a mean from a sample. The larger the sample gets, the smaller the standard error becomes and as the sample size goes to infinity, the SE will tend towards zero. This is different when demographic parameters are known with error. Then, the resulting growth rate will also have an error that does not depend on the number of years chosen in the simulation. You may want to try this out.

Sometimes the temporal variability of the demographic rates is unknown, but we have point estimates from a number of years. In this case we can incorporate into the model the information about this temporal variation in the rates, and incorporate the resulting uncertainty into derived quantities such as the stochastic population growth rate as follows: we randomly pick the value of one year, project the population, and repeat this for a large number of times. For illustration, here is a list with the demographic rates, which are assumed known from 10 years.

```
sj.t <- c(0.303, 0.300, 0.288, 0.310, 0.306, 0.291, 0.303, 0.281, 0.314,
0.303)
sa.t <- c(0.546, 0.545, 0.547, 0.545, 0.547, 0.556, 0.540, 0.551, 0.548,
0.550)
f1.t <- c(1.56, 1.15, 1.38, 1.89, 1.06, 0.69, 1.16, 1.10, 1.48, 1.28)
fa.t <- c(2.03, 1.69, 1.95, 1.98, 1.63, 1.43, 1.63, 1.45, 1.79, 1.91)
```

We create vectors of these demographic rates, such that the number of years corresponds to *T*. We pick years at random first and then construct vectors with the demographic rates.

```
T <- 10000                      # Number of predicted years
t <- 1:10                       # Number of years with actual data
year <- sample(t, T, replace = TRUE)
sj <- sj.t[year]
sa <- sa.t[year]
f1 <- f1.t[year]
fa <- fa.t[year]
```

Since we have used the same vector of selected years for all parameters, any temporal correlations between demographic parameters will be maintained. This is a desired feature of

this simulation method, since any positive correlation between the demographic rates will result in smaller population growth rates while negative correlations tend to dampen fluctuations of the annual growth rates (Tuljapurkar and Orzack 1980, Fieberg and Ellner 2001).

```
# Bundle data
T <- 10000
u <- 100   # Burn-in to calculate the stochastic population growth rate
bugs.data <- list(sj = sj, sa = sa, f1 = f1, fa = fa, T = T, u = u)
```

The analysing code needs a few adaptations. Since the demographic rates are now known for each of the *T* years, we do not need the section with their simulation. Therefore the model does not contain a stochastic node anymore and we need to add a dummy one.

```
# Write BUGS model file
cat(file = "m4.jags", "
model {
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1
dummy ~ dunif(0, 1)      # Arbitrary stochastic node to make JAGS happy

# Loop over time
for (t in 1:T){
   # Population model
   N[1,t+1] <- sj[t] * (f1[t] * N[1,t] + fa[t] * N[2,t])
   N[2,t+1] <- sa[t] * (N[1,t] + N[2,t])

   # Annual (realized) growth rate on log scale
   r.annual[t] <- log(N[1,t+1] + N[2,t+1]) - log(N[1,t] + N[2,t])
   }
r <- mean(r.annual[u:T])
lambda <- exp(r)

# Sensitivity and elasticity of lambda to changes in sj
delta <- 0.001                      # size of perturbation
N.star[1,1] <- 1
N.star[2,1] <- 1
for (t in 1:T){
   N.star[1,t+1] <- (sj[t] + delta) * (f1[t] * N.star[1,t] + fa[t] *
N.star[2,t])
   N.star[2,t+1] <- sa[t] * (N.star[1,t] + N.star[2,t])
   r.annual.star[t] <- log(N.star[1,t+1] + N.star[2,t+1]) - log(N.star[1,t] +
N.star[2,t])
   }
r.star <- mean(r.annual.star[u:T])
s.sj <- (exp(r.star)-lambda) / delta
e.sj <- s.sj * mean(sj) / lambda
}
")


# Initial values
inits <- function(){list(dummy = 0.5)}

# Parameters monitored
parameters <- c("r", "lambda", "s.sj", "e.sj")

# MCMC settings
ni <- 2; nt <- 1; nb <- 0; nc <- 1
```

```
# Call JAGS from R (BRT <1 min)
m4 <- jags(bugs.data, inits, parameters, "m4.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m4, 4)

JAGS output for model 'm4.jags', generated by jagsUI.
Estimates based on 1 chains of 2 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 2 total samples from the joint posterior.
MCMC ran for 0.105 minutes at time 2016-08-16 09:18:50.

          mean sd   2.5%    50%  97.5% overlap0 f
r       0.0046  0 0.0046 0.0046 0.0046    FALSE 1
lambda 1.0046  0 1.0046 1.0046 1.0046    FALSE 1
s.sj    1.4064  0 1.4064 1.4064 1.4064    FALSE 1
e.sj    0.4198  0 0.4198 0.4198 0.4198    FALSE 1
```

The point estimates are similar as before. A main difference is that the standard deviation of all paramaters is now zero. This does make sense, since there is no stochasticity in the model. It would be different if we had generated the sequence of years within the BUGS code. This is possible using the categorical distribution (see exercise XY).

3.4.4. Analysis of a matrix projection model with demographic stochasticity, but without parameter uncertainty

The next kind of stochasticity that we intend to include in the population model analyzed with BUGS is demographic stochasticity. Recall that demographic stochasticity originates from the stochastic nature of demographic processes operating on individuals, and it requires the specification of appropriate discrete-valued distributions that mimick these demographic processes. The resulting population sizes will be discrete and the initial population size matters. As for the classical analysis, we need to project for a number of years the population that is subject to demographic stochasticity. We will obtain from the simulated population trajectories our estimates of the population growth rate, life-history summaries and of extinction probability.

A potential difficulty when fitting models including demographic stochasticity is that population size can drop to zero (i.e., the population may go extinct). Using the equals function in JAGS we can monitor any extinction events: in each year and for each simulation replicate we can check whether the population size at an iteration is equal to zero. Once a population is extinct, we can no longer estimate the population growth rate or sensitivities, because this would involve a division by zero. In these cases BUGS will crash. A way to overcome this problem is to use BUGS only for the population projection and to calculate population growth rate and all life history summaries outside BUGS. This gives us the possibility of using a proper if statement to avoid a division by zero.

```
# Define mean of the demographic parameters
mean.sj <- 0.3
mean.sa <- 0.55
mean.f1 <- 1.3
mean.fa <- 1.8

# Define the number of years with predictions and the Monte Carlo setting
T <- 200

# Define initial stage-specific population sizes
```

```
N1 <- 10
N2 <- 10

# Bundle data
bugs.data <- list(sj = mean.sj, sa = mean.sa, f1 = mean.f1, fa = mean.fa, T =
T, N1 = N1, N2 = N2)

# Write BUGS model file
cat(file = "m5.jags", "
model {
# Model for initial state
N[1,1] <- N1
N[2,1] <- N2

# Loop over time
for (t in 1:T){
   # Population model
   N[1,t+1] ~ dpois(sj * (f1 * N[1,t] + fa * N[2,t]))
   N[2,t+1] ~ dbin(sa, (N[1,t] + N[2,t]))
   extinct[t] <- equals(N[1,t+1] + N[2,t+1], 0)    # Determines whether
population is still thriving (extinct = 0) or went extinct (extinct = 1)
   }
}
")
```

An initial value of at least one stochastic node needs to be given, otherwise BUGS will crash. Population size after the initial year is stochastic (due to demographic stochasticity), hence, we can give initial values for those nodes. In our model, the population size variable is defined as a matrix, thus we need to define a matrix with the same dimension for the initial values, but since population size in the first year is deterministic, no initials must be given then and we therefore set an NA in the first year.

```
# Initial values
Ninit <- matrix(10, nrow = 2, ncol = T + 1)
Ninit[,1] <- NA
inits <- function(){list(N = Ninit)}

# Parameters monitored
parameters <- c("N", "extinct")

# MCMC settings
ni <- 1000; nt <- 1; nb <- 0; nc <- 1
```

We ask for 1,000 simulations of the population. No burn-in is necessary, nor are multiple chains, since convergence is not an issue. This is still not a Bayesian analysis, we still are just using the MCMC algorithm for stochastic simulations, not parameter estimation.

```
# Call JAGS from R
m5 <- jags(bugs.data, inits, parameters, "m5.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m5, 4)

JAGS output for model 'm5.jags', generated by jagsUI.
Estimates based on 1 chains of 1000 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 1000 total samples from the joint posterior.
MCMC ran for 0.013 minutes at time 2016-08-16 09:20:54.

                mean       sd    2.5%    50%    97.5% overlap0 f
```

```
N[1,1]         10.000    0.0000 10.000   10.0    10.000     FALSE 1
N[2,1]         10.000    0.0000 10.000   10.0    10.000     FALSE 1
N[1,2]          9.344    3.0602  4.000    9.0    15.000     FALSE 1
N[2,2]         10.956    2.1975  7.000   11.0    15.000     FALSE 1
N[1,3]          9.391    3.4427  3.000    9.0    16.000     FALSE 1
N[2,3]         11.188    3.0584  6.000   11.0    18.000     FALSE 1
N[1,4]          9.466    3.8732  3.000    9.0    18.000     FALSE 1
[ output truncated … ]
```

We are particularly interested in the estimates of the stochastic population growth rate and the extinction probability. These associated calculations are performed outside of BUGS to avoid the abovementioned calculation problems. This bit of code is very similar to the code for the classical analysis (chapter 3.3.4).

```
# Calculation of the stochastic population growth rate and of the extinction probability outside JAGS
dimensions <- dim(m5$sims.list$extinct)
r.annual <- matrix(NA, nrow = dimensions[2], ncol = dimensions[1])
r <- lambda <- numeric()
for (s in 1:dimensions[1]){
   for (t in 1:dimensions[2]){
     # Calculate annual growth rate on log scale
     if (m5$sims.list$extinct[s,t] == 1) break
     else {
        r.annual[t,s] <- log(m5$sims.list$N[s,1,t+1] +
m5$sims.list$N[s,2,t+1]) - log(m5$sims.list$N[s,1,t] + m5$sims.list$N[s,2,t])
        } # else
     } # t
r[s] <- mean(r.annual[which(m5$sims.list$extinct[s,] == 0),s])
lambda[s] <- exp(r[s])
} # i
```

Now we can easily calculate the mean population growth rate, its precision and the extinction probability as follows:

```
mean(r)
[1] -0.01329435
sd(r)
[1] 0.06259037

# Extinction probability (after T years)
mean(m5$sims.list$extinct[,T])
[1] 0.324
```

We can produce the same plot as Fig. 3.12 (not shown here) and we will notice the same thing: the frequency distribution of the population growth rates is bimodal, because the extinct populations have different growth rates on average than the populations that were still extant after 200 years. We may want to calculate the mean population growth rates of the surviving populations only:

```
mean(r[m5$sims.list$extinct[,T]==0])
[1] 0.01964068
sd(r[m5$sims.list$extinct[,T]==0])
[1] 0.005850182
```

3.4.5. Analysis of a projection matrix model with different sources of stochasticity and with parameter uncertainty

Here we illustrate the combination of the different forms of stochasticity (environmental, demographic) and of parameter uncertainty using MCMC simulations. As in chapter 3.3.5. we assume that we have estimates of the mean and of the temporal variance of the demographic parameter, and that the mean is measured with a known error. The goal is the estimation of the population growth rate and of the extinction probability, in such a way that all uncertainty is propagated adequately.

```
# Define mean, measurement error and temporal variability of the demographic
parameters
mean.sj <- 0.3          # Mean juv. survival on probability scale
sd.sj.e <- 0.005        # Uncertainty of mean juv. survival expressed as SD on
natural scale
sd.sj.t <- 0.25         # Temporal variability of juv. survival expressed as SD
on logit scale
mean.sa <- 0.55         # Mean ad. survival on probability scale
sd.sa.e <- 0.005        # Uncertainty of mean ad. survival expressed as SD on
natural scale
sd.sa.t <- 0.07         # Temporal variability of ad. survival expressed as SD
on logit scale
mean.f1 <- 1.3          # Mean fecundity of 1y females on natural scale
sd.f1.e <- 0.05         # Uncertainty of mean fecundity expressed as SD on
natural scale
sd.f1.t <- 0.3          # Temporal variability of fecundity expressed as SD on
natural scale
mean.fa <- 1.8          # Mean fecundity of adult females on natural scale
sd.fa.e <- 0.03         # Uncertainty of mean fecundity expressed as SD on
natural scale
sd.fa.t <- 0.3          # Temporal variability of fecundity expressed as SD on
natural scale


# Define the number of years with predictions and the Monte Carlo setting
T <- 200

# Bundle data
N1 <- 10
N2 <- 10
bugs.data <- list(mean.sj = mean.sj, mean.sa = mean.sa, mean.f1 = mean.f1,
mean.fa = mean.fa, sd.sj.e = sd.sj.e, sd.sa.e = sd.sa.e, sd.f1.e = sd.f1.e,
sd.fa.e = sd.fa.e, sd.sj.t = sd.sj.t, sd.sa.t = sd.sa.t, sd.f1.t = sd.f1.t,
sd.fa.t = sd.fa.t, T = T, N1 = N1, N2 = N2)


# Write BUGS model file
cat(file = "m6.jags", "
model {
# Priors for the means of the demographic rates on transformed scales,
measurement error added
logit.mean.sj ~ dnorm(log(mean.sj / (1-mean.sj)), tau.logit.sj.e)
logit.mean.sa ~ dnorm(log(mean.sa / (1-mean.sa)), tau.logit.sa.e)
l.mean.f1 ~ dnorm(mean.f1, tau.f1.e)
l.mean.fa ~ dnorm(mean.fa, tau.fa.e)

# Calculate precision of the measurement error of demographic rates
tau.logit.sj.e <- pow(sd.sj.e, -2)
tau.logit.sa.e <- pow(sd.sa.e, -2)
```

```
tau.f1.e <- pow(sd.f1.e, -2)
tau.fa.e <- pow(sd.fa.e, -2)

# Calculate precision for the temporal variability of demographic rates
tau.logit.sj.t <- pow(sd.sj.t, -2)
tau.logit.sa.t <- pow(sd.sa.t, -2)
tau.f1.t <- pow(sd.f1.t, -2)
tau.fa.t <- pow(sd.fa.t, -2)

# Priors for the annual demographic rates
for (t in 1:T){
   sj[t] <- 1/(1+exp(-logit.sj[t]))      # backtransformation from logit scale
   logit.sj[t] ~ dnorm(logit.mean.sj, tau.logit.sj.t)
   sa[t] <- 1/(1+exp(-logit.sa[t]))      # backtransformation from logit scale
   logit.sa[t] ~ dnorm(logit.mean.sa, tau.logit.sa.t)
   f1[t] ~ dnorm(l.mean.f1, tau.f1.t)
   fa[t] ~ dnorm(l.mean.fa, tau.fa.t)
   }

# Model for initial state
N[1,1] <- N1
N[2,1] <- N2

# Loop over time
for (t in 1:T){
   # Population model
   N[1,t+1] ~ dpois(sj[t] * (f1[t] * N[1,t] + fa[t] * N[2,t]))
   N[2,t+1] ~ dbin(sa[t], (N[1,t] + N[2,t]))
   extinct[t] <- equals(N[1,t+1] + N[2,t+1], 0)    # Determines whether
population is still thriving (extinct = 0) or went extinct (extinct = 1)
   } # t
}
")



# Initial values
inits <- function(){list(logit.mean.sj = qlogis(0.3), logit.mean.sa =
qlogis(0.55))}

# Parameters monitored
parameters <- c("N", "extinct")

# MCMC settings
ni <- 1000; nt <- 1; nb <- 0; nc <- 1

# Call JAGS from R and summarize simulation
m6 <- jags(bugs.data, inits, parameters, "m6.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m6, 4)

JAGS output for model 'm6.jags', generated by jagsUI.
Estimates based on 1 chains of 1000 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 1000 total samples from the joint posterior.
MCMC ran for 0.026 minutes at time 2016-08-16 09:45:08.

                mean         sd   2.5%   50%     97.5% overlap0 f
N[1,1]        10.000     0.0000 10.000  10.0    10.000    FALSE 1
N[2,1]        10.000     0.0000 10.000  10.0    10.000    FALSE 1
N[1,2]         9.417     3.8160  3.000   9.0    18.000    FALSE 1
N[2,2]        11.065     2.2561  6.000  11.0    15.000    FALSE 1
N[1,3]         9.580     4.2125  2.000   9.0    19.000    FALSE 1
N[2,3]        11.241     3.3380  6.000  11.0    18.000    FALSE 1
```

```
N[1,4]         10.068     4.9514  2.000   9.0    22.000    FALSE 1
N[2,4]         11.409     4.0795  4.975  11.0    20.000    FALSE 1
N[1,5]         10.071     5.4071  2.000   9.0    23.000    FALSE 1
N[2,5]         11.844     4.7919  4.000  11.0    23.000    FALSE 1
```

```
# Calculation of the stochastic population growth rate outside JAGS
dimensions <- dim(m6$sims.list$extinct)
r.annual <- matrix(NA, nrow = dimensions[2], ncol = dimensions[1])
r <- lambda <- numeric()
for (s in 1:dimensions[1]){
   for (t in 1:dimensions[2]){
     # Calculate annual growth rate on log scale
     if (m6$sims.list$extinct[s,t] == 1) break
     else {
        r.annual[t,s] <- log(m6$sims.list$N[s,1,t+1] +
m6$sims.list$N[s,2,t+1]) - log(m6$sims.list$N[s,1,t] + m6$sims.list$N[s,2,t])
        } # else
     } # t
r[s] <- mean(r.annual[which(m6$sims.list$extinct[s,] == 0),s])
lambda[s] <- exp(r[s])
} # i

mean(r)
[1] -0.01288431

# Population growth rate of populations that did not went extinct
mean(r[m6$sims.list$extinct[,T]==0])
[1] 0.02129162

sd(r[m6$sims.list$extinct[,T]==0])
[1] 0.01046524

# Extinction probability (after T years)
mean(m6$sims.list$extinct[,T])
[1] 0.362
```

3.4.6. Projection models with density-dependence and demographic stochasticity

The code for a model with density-dependence and demographic stochasticity poses no particular challenge. We want to obtain 1000 simulation samples.

```
# Define means of the demographic rates and strength of density-dependence
mean.sj <- 0.3
mean.sa <- 0.55
f1.int <- 2.3           # Fecundity of 1y females when population size is 0
f1.beta <- -0.02        # Strength of density-dependence on fecundity
fa.int <- 2.3           # Fecundity of adult females when population size is 0
fa.beta <- -0.01        # Strength of density-dependence on fecundity

# Define the number of years with predictions
T <- 200

# Bundle data
bugs.data <- list(sj = mean.sj, sa = mean.sa, f1.int = f1.int, f1.beta =
f1.beta, fa.int = fa.int, fa.beta = fa.beta, T = T)


# Write BUGS model file
cat(file = "m7.jags", "
```

```
model {
# Model for initial state
N[1,1] <- 10
N[2,1] <- 10

# Loop over time
for (t in 1:T){
   # Calculate actual productivity
   f1[t] <- f1.int + f1.beta * (N[1,t] + N[2,t])
   fa[t] <- fa.int + fa.beta * (N[1,t] + N[2,t])

   # Population model
   N[1,t+1] ~ dpois(sj * (f1[t] * N[1,t] + fa[t] * N[2,t]))
   N[2,t+1] ~ dbin(sa, (N[1,t] + N[2,t]))
   extinct[t] <- equals(N[1,t+1] + N[2,t+1], 0)    # Determines whether
population is still thriving (extinct = 0) or went extinct (extinct = 1)
   }
}
")


# Initial values
N <- matrix(NA, nrow = 2, ncol = T+1)
N[,2:(T+1)] <- 10
inits <- function(){list(N = N)}

# Parameters monitored
parameters <- c("N", "f1", "fa", "extinct")

# MCMC settings
ni <- 1000; nt <- 1; nb <- 0; nc <- 1

# Call JAGS from R and summarize simulations
m7 <- jags(bugs.data, inits, parameters, "m7.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb, DIC = FALSE)
print(m7, 4)

JAGS output for model 'm7.jags', generated by jagsUI.
Estimates based on 1 chains of 1000 iterations,
burn-in = 0 iterations and thin rate = 1,
yielding 1000 total samples from the joint posterior.
MCMC ran for 0.053 minutes at time 2016-08-16 10:35:01.

                mean     sd    2.5%    50%   97.5% overlap0 f
N[1,1]       10.0000 0.0000 10.0000 10.000 10.0000    FALSE 1
N[2,1]       10.0000 0.0000 10.0000 10.000 10.0000    FALSE 1
N[1,2]       11.8560 3.4291  6.0000 12.000 19.0000    FALSE 1
N[2,2]       11.0660 2.2814  6.0000 11.000 15.0000    FALSE 1
N[1,3]       13.5010 4.1835  6.0000 13.000 22.0000    FALSE 1
N[2,3]       12.7190 3.3138  7.0000 12.000 20.0000    FALSE 1
N[1,4]       14.7750 4.8141  6.0000 14.000 25.0000    FALSE 1
 [output truncated... ]
f1[1]         1.9000 0.0000  1.9000  1.900  1.9000    FALSE 1
f1[2]         1.8416 0.0826  1.6800  1.840  2.0000    FALSE 1
f1[3]         1.7756 0.1211  1.5400  1.780  2.0200    FALSE 1
f1[4]         1.7186 0.1504  1.4400  1.720  1.9800    FALSE 1
f1[5]         1.6657 0.1767  1.3200  1.680  2.0000    FALSE 1
[output truncated... ]
fa[1]         2.1000 0.0000  2.1000  2.100  2.1000    FALSE 1
fa[2]         2.0708 0.0413  1.9900  2.070  2.1500    FALSE 1
fa[3]         2.0378 0.0605  1.9200  2.040  2.1600    FALSE 1
fa[4]         2.0093 0.0752  1.8700  2.010  2.1400    FALSE 1
fa[5]         1.9829 0.0883  1.8100  1.990  2.1500    FALSE 1
```

A graph similar to Fig. 3.14 could again be produced, but we do not show this. Instead we compute the carrying-capacity, average fecundities at carrying-capacity and the extinction probability

```
mean(rowSums(m7$sims.list$N[,,T+1]))
[1] 53.89

mean(m7$sims.list$f1[,200])
[1] 1.22508
mean(m7$sims.list$fa[,200])
[1] 1.76254


mean(m7$sims.list$extinct[,T] == 1)
[1] 0
```

All these estimates are very similar to those obtained with classical simulation methods (chapter 3.3.6).

5. Introduction to integrated population models

5.1. Introduction

In this chapter we will progressively develop our first integrated population model. We will start by the use of capture-recapture data to estimate survival probabilities that are then applied in an population projection model to estimate the asymptotic population growth rate and its precision. This is not yet an integrated analysis, because we still use a single data set (capture-recapture data), and the estimate of the population growth rate from the projection model can be seen as a derived parameter. Second, we will in addition include productivity data and again estimate the asymptotic population growth rate and its precision. Now, the analysis is a joint analysis, because we analyse two data sets simultaneously. Third, we will also include population count data. The state-space model for the count data is formulated in terms of the same demographic rates that we get from the demographic data and thus the resulting model is now a truely integrated model, because information about some parameters stem from different sources.

We stress that integrated population models are an extension of matrix projection models in the sense that models for the analyses of demographic and count data are combined with matrix projection models. It will become obvious that we can derive the same kind of quantities (life history summaries) and perform the same kind of perturbation analysis from integrated population model as we can do from matrix projection models. Because all integrated population models as used (defined) in this book countain count data or estimated population indices that are analysed with state-space models, we can also say that integrated population models are state-space models that are coupled with model for demographic data.

In the rest of the chapter we have a closer look at our first integrated population model. The joint likelihood is derived and we discuss assumptions of the model. Finally, we experience advatages of intergared population models in terms of increased precision of parameter estimates and in terms of parameter estimability.

5.2. Use of demographic data in the analysis of a population projection model

5.2.1. Combining capture-recapture data with a population projection model

We still use our woodchat shrike example that we have introduced in chapter 3. We assume that we have sampled capture-recaptue data during 10 years, and that the underlying survival probabilities are 0.3 for juveniles and 0.55 for the adults. Each year we have marked 50 juveniles and 15 adults. The recapture probability is assumed to be 0.6. We assume in addition that the woodchat shrikes do not disperse outside the study area, thus the estimated survival probabilities are not apparent, they are true and the population model does not need adaptations regarding dispersal. This is of course an assumption that is typically violated and unrealistic in many cases. In a later chapter we will show how one can deal with open populations and capture-recapture models (Chapter XY).

We simulate capture-recapture data using the data generating function as shown in chapter 5.X. The use of the multinomial likelihood is preferred over the state-space likelihood, because the former runs must faster. Therefore, we also need the function that creates the m-array from the capture-recapture data.

```
# Define function to simulate a capture-history (CH) matrix
simul.cjs <- function(PHI, P, marked){
   n.occasions <- dim(PHI)[2] + 1
```

```
   CH <- matrix(0, ncol = n.occasions, nrow = sum(marked))
   # Define a vector with the occasion of marking
   mark.occ <- rep(1:length(marked), marked[1:length(marked)])
   # Fill the CH matrix
   for (i in 1:sum(marked)){
      CH[i, mark.occ[i]] <- 1       # Write an 1 at the release occasion
      if (mark.occ[i]==n.occasions) next
      for (t in (mark.occ[i]+1):n.occasions){
         # Bernoulli trial: does individual survive occasion?
         sur <- rbinom(1, 1, PHI[i,t-1])
         if (sur==0) break          # If dead, move to next individual
         # Bernoulli trial: is individual recaptured?
         rp <- rbinom(1, 1, P[i,t-1])
         if (rp==1) CH[i,t] <- 1
         } #t
      } #i
   return(CH)
   }
```

 # Function to create age-dependent m-arrays from single state capture-histories

```
marray.age <- function(ch, age, mAge = 1){

# Input variables
#    ch: matrix with capture histories.
#        Note: the capture history file is a single file that includes the
individuals of all age classes
#    age: vector with the age class at first capture for each individual
#    mAge: maximal number of age classes for which m-arrays are constructed.
Input is optional and only required if the age matrix has fewer age classes as
we want to separate (e.g. CH contains only individuals marked as juveniles,
and we want 2 age classes)
#
# Output
#    marr: 3-d array with the m-array. The third dimension is the age class.
The last column of each m-array is the number of released individuals that
were never recaptured. Thus, the total number of released individuals per
occasion is the row sum of the m-array.

   # 1. Define subfunctions
   # 1.1. Function to create a m-array from capture-histories (ch)
  marray <- function(ch, unobs = 0){
      ns <- length(table(ch)) - 1 + unobs
      no <- ncol(ch)
      out <- matrix(0, ncol = ns*(no-1)+1, nrow = ns*(no-1))
      # Remove capture histories of ind. that are marked at last occasion
      get.first <- function(x) min(which(x!=0))
      first <- apply(ch, 1, get.first)
      last <- which(first==no)
      if (length(last) > 0) ch <- ch[-last,]
      # Compute m-array
      for (i in 1:nrow(ch)){
         cap.occ <- which(ch[i,]!=0)
         state <- ch[i,cap.occ]
         if (length(state) == 1) {
            out[state[1]+ns*(cap.occ[1]-1), ns*(no-1)+1] <-
out[state[1]+ns*(cap.occ[1]-1), ns*(no-1)+1] + 1
            }
         if (length(state) > 1) {
            for (t in 2:length(cap.occ)){
               out[(cap.occ[t-1]-1)*ns+state[t-1], (cap.occ[t]-2)*ns+state[t]]
<- out[(cap.occ[t-1]-1)*ns+state[t-1], (cap.occ[t]-2)*ns+state[t]] + 1
               } # t
```

```
            if (max(cap.occ) < no){
               out[(cap.occ[t]-1)*ns+state[t], ns*(no-1)+1] <-
out[(cap.occ[t]-1)*ns+state[t], ns*(no-1)+1] + 1
               } # if
            } # if
         } # t
      return(out)
      }

   # 1.2. Function to remove capture histories without any capture
   clean.ch <- function(ch){
      incl <- which(rowSums(ch)>=1)
      ch <- ch[incl,]
      return(ch)
      }

   # 1.3. Function to remove all first captures from the capture-histories
   rm.first <- function(ch) {
      get.first <- function(x) min(which(x==1))
      first <- apply(ch, 1, get.first)
      for (i in 1:nrow(ch)){
         ch[i,first[i]] <- 0
         }
      return(ch)
      }

   # 1.4. Function to calculate the occasion of first capture
   get.first <- function(x) min(which(x==1))


   # 2. Calculations
   if (is.matrix(ch)==FALSE) ch <- matrix(ch, nrow = 1)
   maxAge <- max(c(max(age), mAge))
   nind <- nrow(ch)
   n.occasions <- ncol(ch)

   first <- apply(ch, 1, get.first)
   age.matrix <- matrix(0, ncol = n.occasions, nrow = nind)
   for (i in 1:nind){
      age.matrix[i,first[i]:n.occasions] <- 1:(n.occasions-
first[i]+1)+(age[i]-1)
      }
   age.matrix[age.matrix > maxAge] <- maxAge

   # Recode capture history
   ch.rec <- ch
   for (i in 1:nind){
      h <- which(ch.rec[i,]==1)
      for (j in 1:length(h)){
         ch.rec[i,h[j]] <- j
         } # j
      } # i
   ch.rec[ch.rec > maxAge] <- maxAge

   ch.split <- array(0, dim = c(nrow(ch), ncol(ch), maxAge))
   for (a in 1:maxAge){
      for (i in 1:nind){
         j <- which(ch.rec[i,]==a | ch.rec[i,]==(a+1))
         if (length(j)==0) next
         ch.split[i,j[1:2],age.matrix[i,j[1]]] <- 1
         if (length(j)>1){
            ch.split[i,j[2:length(j)],age.matrix[i,j[2]]] <- 1
            }
```

```
      } # i
   } # a

marr <- array(0, dim = c(n.occasions-1, n.occasions, maxAge))
for (a in 1:(maxAge-1)){
   for (i in 1:nind){
      u <- which(ch.split[i,,a]==1)
      if (length(u)==0) next
      if (u[1]==n.occasions) next
      if (length(u)==1) marr[u,n.occasions,a] <- marr[u,n.occasions,a] + 1
      if (length(u)==2) marr[u[1],u[2]-1,a] <- marr[u[1],u[2]-1,a] + 1
      } # i
   } # a
a <- maxAge

if (is.matrix(ch.split[,,a])==FALSE){
   ch.split1 <- matrix(ch.split[,,a], nrow = 1)
   marr[,,a] <- marray(ch.split1)
   } # if
else marr[,,a] <- marray(ch.split[,,a])
return(marr)
}
```

```
# Simulation of capture-recapture data
# Define parameter values
n.occasions <- 10                    # Number of capture occasions
marked.j <- rep(50, n.occasions-1)   # Annual number of newly marked juveniles
marked.a <- rep(20, n.occasions-1)   # Annual number of newly marked adults
phi.juv <- 0.3                       # Juvenile annual survival
phi.ad <- 0.55                       # Adult annual survival
p <- rep(0.6, n.occasions-1)         # Recapture
phi.j <- c(phi.juv, rep(phi.ad,n.occasions-2))
phi.a <- rep(phi.ad, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI.J <- matrix(0, ncol = n.occasions-1, nrow = sum(marked.j))
for (i in 1:length(marked.j)){
   PHI.J[(sum(marked.j[1:i])-marked.j[i]+1):sum(marked.j[1:i]),i:(n.occasions-
1)] <- matrix(rep(phi.j[1:(n.occasions-i)],marked.j[i]), ncol = n.occasions-i,
byrow = TRUE)
   }
P.J <- matrix(rep(p, n.occasions*sum(marked.j)), ncol = n.occasions-1, nrow =
sum(marked.j), byrow = TRUE)
PHI.A <- matrix(rep(phi.a, sum(marked.a)), ncol = n.occasions-1, nrow =
sum(marked.a), byrow = TRUE)
P.A <- matrix(rep(p, sum(marked.a)), ncol = n.occasions-1, nrow =
sum(marked.a), byrow = TRUE)

# Apply simulation function
CH.J <- simul.cjs(PHI.J, P.J, marked.j)
CH.A <- simul.cjs(PHI.A, P.A, marked.a)

# Create m-arrays
CH <- rbind(CH.J, CH.A)
age <- c(rep(1, nrow(CH.J)), rep(2, nrow(CH.A)))
shrike.marray <- marray.age(CH, age)
```

The goal in our example is the estimation of the asymptotic population growth rate (λ) and its precision, by using the survival probabilities that we have estimated and a value of productivity that originates from the literature (*f* = 1.6). To develop the needed code we take the code for the estimation of the asymptotic population growth rate, where we had the

possibility to include uncertainty about the demographic parameters (chapter 3.4.1). Originally, we added information about survival using a distribution (which we called “prior” although in the statistical sense it was not a prior), from where values of survival were generated. Now we add information about survival originating from the capture-recapture data and hence we add code that estimates age-specific survival from the m-array data (chapter 5.XY). We still need to define priors for the survival parameters in the combined code, but this time they can be uninformative (vague) – and, now they are true priors, because we estimate a quantity (survival) from data. The order in which we combine the two bits of code (matrix projection model and capture-recapture model) is irrelevant.

```
# Specify the model in BUGS language
cat(file = "m1.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

for (t in 1:(n.occasions-1)){
   sj[t] <- mean.sj
   sa[t] <- mean.sa
   p[t] <- mean.p
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]            # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
   pr.a[t,t] <- sa[t]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j])*prod(q[t:(j-1)])*p[j]
      pr.a[t,j] <- prod(sa[t:j])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Population model
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1

# Loop over time
```

```
for (t in 1:T){
   # Population projection
   N[1,t+1] <- f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   # Annual growth rate
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
lambda <- ann.growth.rate[T]
}
")
```

The data consist of the sampled m-arrays, the information about productivity (that we assume to know without measurement error) and the number of years for which we project the population to get an estimate of λ.

```
# Bundle data
bugs.data <- list(marr.j = shrike.marray[,,1], marr.a = shrike.marray[,,2],
n.occasions = dim(shrike.marray)[2], rel.j = rowSums(shrike.marray[,,1]),
rel.a = rowSums(shrike.marray[,,2]), f = 1.6, T = 20)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 1), mean.sa = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.p", "lambda")
```

In the MCMC settings we include now a burn-in, because we now "truly" estimate parameters (the survival probabilities) and we have to make sure that their estimations have converged. To check convergence we also use multiple chains.

```
# MCMC settings
ni <- 3000; nt <- 1; nb <- 1000; nc <- 3

# Call JAGS from R (jagsUI)
m1 <- jags(bugs.data, inits, parameters, "m1.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m1, 3)

JAGS output for model 'm1.jags', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran for 0.027 minutes at time 2016-06-09 11:36:45.

            mean    sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj    0.309 0.027   0.258   0.308   0.364    FALSE 1 1.000  6000
mean.sa    0.555 0.025   0.505   0.554   0.604    FALSE 1 1.003   788
mean.p     0.619 0.035   0.548   0.620   0.686    FALSE 1 1.001  1478
lambda     1.048 0.051   0.953   1.047   1.152    FALSE 1 1.001  3007
deviance 181.852 2.434 179.104 181.217 188.161    FALSE 1 1.004  1088
```

Parameter convergence looks satisfactory and the estimates of the survival and recapture probabilities correspond well with the values to simulate the capture histories. The asymptotic population growth rate is slightly larger than the one that we obtained previously. This is because both survival probabilities are slightly higher than the data generating values. It is of course possible to compute also other life history summaries such as growth rate

sensitivities, the stable age ratio or generation time; all of them are derived quantities and their computation can be included in the code as shown in chapter 3.4.1.

5.2.2. Combining capture-recapture and productivity data with a population projection model

We have assumed that productivity is known without measurement error, but we could of course also include uncertainty about this parameter. Again, this can be done in exactly the same way as shown in chapter 3.4.2. However, we now assume that we have sampled in addition data about productivity and include data and parameter estimation in the model. The analysing model for the productivity data is a simple Poisson regression model. We then combine this bit of code with the code that we have developed in 6.2.1.

We first simulate data about productivity. We assume that we have information about the number of fledglings from 200 nests. Thus, we create a vector of length 200 with the number of fledglings of each surveyed nest.

```
# Simulation of productivity data
mean.f <- 1.6
J <- rpois(200, mean.f)
```

The inclusion of the productivity data only needs slight adaptations of the model code. Basically a Poisson regression model is added to analyse the productivity data and the estimate of mean productivity is used in the population model. The prior for mean productivity can now be vague, since information about productivity come from the data.

```
# Specify the model in BUGS language
cat(file = "m2.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

for (t in 1:(n.occasions-1)){
   sj[t] <- mean.sj
   sa[t] <- mean.sa
   p[t] <- mean.p
   }

# Poisson regression model for productivity data
for (i in 1:n.J){
   J[i] ~ dpois(mean.f)
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]            # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
```

```
   pr.a[t,t] <- sa[t]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j])*prod(q[t:(j-1)])*p[j]
      pr.a[t,j] <- prod(sa[t:j])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Population model
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1

# Loop over time
for (t in 1:T){
   # Population projection
   N[1,t+1] <- mean.f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   # Annual growth rate
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
lambda <- ann.growth.rate[T]
}
")

# Bundle data
bugs.data <- list(marr.j = shrike.marray[,,1], marr.a = shrike.marray[,,2],
n.occasions = dim(shrike.marray)[2], rel.j = rowSums(shrike.marray[,,1]),
rel.a = rowSums(shrike.marray[,,2]), J = J, n.J = length(J), T = 20)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 1), mean.sa = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.p",  "mean.f", "lambda")

# MCMC settings
ni <- 3000
nt <- 1
nb <- 1000
nc <- 3

# Call JAGS from R (jagsUI)
m2 <- jags(bugs.data, inits, parameters, "m2.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m2, 3)

JAGS output for model 'm2.jags', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
```

```
MCMC ran for 0.039 minutes at time 2016-06-09 11:37:09.

            mean    sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj    0.309 0.029   0.255   0.308   0.367    FALSE 1 1.001  4738
mean.sa    0.554 0.025   0.505   0.554   0.605    FALSE 1 1.000  6000
mean.p     0.620 0.035   0.548   0.620   0.688    FALSE 1 1.001  1328
mean.f     1.517 0.086   1.350   1.513   1.692    FALSE 1 1.000  6000
lambda     1.023 0.056   0.919   1.021   1.136    FALSE 1 1.001  3071
deviance 804.069 2.880 800.535 803.430 811.482    FALSE 1 1.001  6000
```

The estimate of productivity is slightly lower than the data generation parameter in this trial, but still correspond quite well. Consequently the asymptotic population growth rate gets smaller than before. As expected is the coefficient of variation of the asymptotic population growth rate now larger than before when we assumed the absence of a measurement error for fecundity (coefficients of variation 0.055 and 0.049, respectively).

We have now jointly analysed capture-recapture and productivity data and included them into a population model to estimate the asymptotic population growth rate, thus we performed a joint analysis. The uncertainties due to data sampling (estimation errors) are correctly and automatically propagated into the error of the asymptotic population growth rate. The classical way to estimate the population growth rate using the two data sates is to first analyse each of the sampled data sets separately to obtain estimates of the mean and the standard deviation of survival and fecundity. In a second step a population projection model is set up and Monte Carlo simulations performed to get the mean and a correct estimate of uncertainty of the population growth rate as shown in chapter 3.3.2. Using a joint analysis the error propagation occurs automatically, that's great, is it?

5.3. The first integrated population model

As a final step towards the first integrated population model we also assume that we have collected count data from the population (population survey) and include them into the model. The count data are analysed with a state-spece model (chapter 4), which is composed of a process and an observation model. The process model corresponds exactly to our matrix projection model, and thus, we basically have to add an observation model to the code that we have developed previously.

Before we can start with the development of the code for our first integrated population model we simulate the necessary data. Because we want to collect count data from the population, we have to generate a population of shrikes that is subject to the predefined underlying demographic rates. We then sample all the data sets we like from this population. The population generation function that we have written simulates the fate of all individuals in the population. This has the advantage that demographic stochasticity is automatically included and that any kind of data sampling (e.g. capture-recapture, productivity, counts) can be applied. The population generation function function needs as input the number of years, the stage- and year-specific demographic rates and the stage-specific population sizes at the beginning of the study (t = 1).

Here we define the parameters that are needed for the population generation function:

```
# Age specific survival probabilities (juv, adult)
sj <- 0.3
sa <- 0.55

# Fecundity rate (females)
f1 <- 1.6            # productivity of 1 year old females
```

```
f2 <- f1              # productivity of females older than one year

# Initial population size per age class
Ni <- c(100, 100)

# Number of years
T <- 10
```

We apply function `create.population` (see appendix) to create the shrike population:

```
ind <- create.population(phi = matrix(c(rep(sj, T-1), rep(sa, T-1)), ncol = T-
1, byrow = TRUE), f = matrix(c(rep(f1, T), rep(f2, T)), ncol = T, byrow =
TRUE), Im = rep(0, T), Ni = Ni, seed = 2008)
```

The object `ind` contains information about the life (i.e. when they were born or have arrived in the population as immigrant, when they have died, when they have produced how many offspring) of all females in the population as well as a summary table that lists the stage-specific numbers of individuals in each year. From this population we can sample the three different kind of demographic data.

Regarding population survey data we assume having counted each year how many breeding pairs of woodchat shrike were present in our study area. We therefore assume a value for the observation error that is the same in each year and use the Normal distribution to create the survey data. Since the generated counts are not integers, we may round them to the nearest integer.

```
# Observation error for the population survey
sigma <- 10

# Create the population survey data
count <- round(create.survey.norm(ind$Nu["Total",], rep(sigma, T), seed = 1))
```

You may not be happy with the way how we have here generated the population count data because we assumed that the counts are correct on average and that counts thus contain errors because of individuals that were missed as well as errors due to doublecounting. You may claim that often only the overlooking of individuals is an issue (i.e. only false negatives) and that the binomial distribution would be the more adequate one for generating count data. Admittedly, the applied Normal distribution may not always be reasonable in reality, but we stick on this assumption for the moment so that we can use exactly the same state-space models as we have introduced in chapter 5. We will see later other observation models for the count data.

Next we create the capture-recapture data and assume that our sampling is not perfect. We need to define the capture probability for the juveniles and the adults, although this parameter is not estimated in the model. This has to be done to avoid the unreastic assumption that all individuals in the population are marked. The recapture probability, that becomes a model parameter, also needs to be specified and here we assume that it is higher than the initial capture probability. From the created capture-histories we construct the age-dependent m-arrays.

```
# Capture and recapture probabilities
cjuv <- 0.3           # initial capture probability of juveniles
cad <- 0.3            # initial capture probability of adults
prec <- 0.6           # recapture probability

# Create the capture histories and the corresponding m-arrays
```

```
ch <- create.capturehistory(ind$IND, c = matrix(c(rep(cjuv, T), rep(cad, T)),
nrow = 2, byrow = TRUE), p = matrix(c(rep(prec, T-1), rep(prec, T-1)), nrow =
2, byrow = TRUE), seed = 1)
marray <- marray.age(ch$ch, ch$age)
```

Finally data on productivity are generated. Here we assume that a nest is detected with a certain probability and if the nest is found, its output (i.e. the number of fledglings) is recorded without error.

```
# Probability to find a brood whose reproductive ouput is recorded
pprod <- 0.3

# Create productivity data
P <- create.reproduction(ind$IND, rep(pprod, T), seed = 1)
```

The productivity data are constructed in two formats, both are stored in the object `P`. The first one contains the individual data where the output of each brood is recorded along with the year in which is it recorded and the age of the mother. The second object includes aggregated data where to total number of fledglings and the total number of surveyed broods are recorded in each year. We will use the individual data here.

Since we assumed that none of the samplings were perfect not all individuals will be included in each data set. However, we have to note that these three samples are not completely independent from each other, because it is very likely that some individuals appear in more than one data set. The risk of having dependent data is reduced when the capture, sighting or nest detection probabilities are low. Since we will analyse the data as if they are independent, the so-called independence assumption is violated. The consequence of this violation using the kind of data and model as here is typically of minor importance (Abadi et al. 2010a), but may be more substantial in other situations (Besbeas et al. 2009). More on the independence assumption, the effects of its violation and how to take non-independence into account is given in chapter XY.

The model code to analyse these data is still similar to what we have seen before. The main difference is that the previous matrix projection model becomes the process model of the state-space model and an observation model for the count data is added. Furthermore, we also replace the fixed stage-specific population sizes in the first year by priors, because we want to estimate these quantities as well. Since the count in the first year is around 200 (sum of both stage classes) but we lack more specific information, we specify uniform priors in the range 1 to 300 for each of the two stage classes.

The annual population growth rates can also be calculated. Since the model contains neither environmental nor demographic stochasticity and a stable stage distribution is obtained quickly with the life history of the woodchat shrike (see chapter 3), the annual population growth rates are after a short transient phase identical to the asymptotic population growth rate. Yet, this is specific to this example and not generally true. We have included a section with "derived" parameters in the code where the annual population growth rates are computed to highlight that this calculation is not a necessary part of the integrated population model.

```
# Specify the model in BUGS language
cat(file = "m3.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
```

```
mean.f ~ dunif(0, 10)

for (t in 1:(n.occasions-1)){
   sj[t] <- mean.sj
   sa[t] <- mean.sa
   p[t] <- mean.p
   }

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# State-space model for count data
# Model for the initial population size: discrete uniform priors
N[1,1] ~ dunif(1, 300)
N[2,1] ~ dunif(1, 300)

# Process model over time
for (t in 1:(n.occasions-1)){
   N[1,t+1] <- mean.f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t] + N[2,t], tau.obs)
   }

# Poisson regression model for productivity data
for (i in 1:n.J){
   J[i] ~ dpois(mean.f)
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]   # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
   pr.a[t,t] <- sa[t]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j])*prod(q[t:(j-1)])*p[j]
      pr.a[t,j] <- prod(sa[t:j])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Derived parameters
# Annual population growth rate
```

```
for (t in 1:(n.occasions-1)){
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
# Total population size
for (t in 1:n.occasions){
   Ntot[t] <- N[1,t] + N[2,t]
   }
}
")
```

```
# Bundle data
bugs.data <- list(marr.j = marray[,,1], marr.a = marray[,,2], n.occasions = T,
rel.j = rowSums(marray[,,1]), rel.a = rowSums(marray[,,2]), J = P$rep.ind[,1],
n.J = nrow(P$rep.ind), count = count)
```

The specification of appropriate initial values for integrated population models can be a pain (see later), but in this simple example it is easy: it is enough to give initials just for the two survival probabilities.

```
# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 0.5), mean.sa = runif(1, 0,
1))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.p", "mean.f", "N", "sigma.obs",
"ann.growth.rate", "Ntot")

# MCMC settings
ni <- 12000; nt <- 6; nb <- 2000; nc <- 3

# Call JAGS from R (jagsUI)
m3 <- jags(bugs.data, inits, parameters, "m3.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m3, 3)

JAGS output for model 'm3.jags', generated by jagsUI.
Estimates based on 3 chains of 12000 iterations,
burn-in = 2000 iterations and thin rate = 6,
yielding 5001 total samples from the joint posterior.
MCMC ran for 0.217 minutes at time 2016-06-20 11:35:39.

                      mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
mean.sj              0.297  0.013    0.271    0.297    0.324    FALSE 1 1.002  2046
mean.sa              0.552  0.019    0.513    0.552    0.588    FALSE 1 1.002  3101
mean.p               0.665  0.027    0.611    0.665    0.716    FALSE 1 1.000  5001
mean.f               1.615  0.046    1.529    1.614    1.709    FALSE 1 1.000  5001
N[1,1]              97.754 55.265    5.423  100.632  186.805    FALSE 1 1.041    54
N[2,1]              92.547 55.487    5.375   87.704  188.701    FALSE 1 1.039    57
N[1,2]              91.124  5.301   80.716   91.001  101.632    FALSE 1 1.000  5001
N[2,2]             104.953  6.513   91.892  105.044  117.767    FALSE 1 1.002  1134
N[1,3]              93.904  5.015   84.194   93.796  103.767    FALSE 1 1.000  5001
N[2,3]             108.143  6.039   96.125  108.223  119.966    FALSE 1 1.002  1175
N[1,4]              96.777  4.840   87.446   96.686  106.364    FALSE 1 1.000  5001
N[2,4]             111.441  5.646  100.088  111.481  122.624    FALSE 1 1.002  1283
N[1,5]              99.748  4.818   90.489   99.699  109.308    FALSE 1 1.001  4217
N[2,5]             114.849  5.384  104.108  114.823  125.471    FALSE 1 1.002  1533
N[1,6]             102.819  4.981   93.434  102.792  112.812    FALSE 1 1.001  2992
N[2,6]             118.373  5.305  107.795  118.417  128.789    FALSE 1 1.001  2100
N[1,7]             105.994  5.346   95.910  106.015  116.791    FALSE 1 1.001  2290
N[2,7]             122.016  5.459  111.124  122.091  132.726    FALSE 1 1.001  3492
N[1,8]             109.277  5.908   98.247  109.282  121.303    FALSE 1 1.001  1905
N[2,8]             125.783  5.867  113.819  125.803  137.438    FALSE 1 1.000  5001
```

```
N[1,9]              112.672  6.653  100.090  112.631  126.055    FALSE 1 1.001  1693
N[2,9]              129.678  6.525  116.565  129.660  142.513    FALSE 1 1.000  5001
N[1,10]             116.184  7.562  102.129  116.113  131.451    FALSE 1 1.001  1575
N[2,10]             133.705  7.409  119.026  133.699  148.177    FALSE 1 1.000  5001
sigma.obs            18.997  5.610   11.349   17.806   33.151    FALSE 1 1.000  5001
ann.growth.rate[1]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[2]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[3]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[4]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[5]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[6]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[7]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[8]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
ann.growth.rate[9]    1.031  0.010    1.012    1.031    1.051    FALSE 1 1.001  1865
Ntot[1]             190.301 10.625  168.927  190.282  211.159    FALSE 1 1.001  1813
Ntot[2]             196.077  9.395  177.139  196.249  214.289    FALSE 1 1.001  1875
Ntot[3]             202.047  8.223  185.464  202.183  217.715    FALSE 1 1.001  2035
Ntot[4]             208.218  7.209  193.760  208.383  222.160    FALSE 1 1.001  2446
Ntot[5]             214.597  6.518  201.643  214.627  227.428    FALSE 1 1.000  3617
Ntot[6]             221.192  6.364  208.305  221.182  234.003    FALSE 1 1.000  5001
Ntot[7]             228.010  6.896  214.043  228.040  241.712    FALSE 1 1.000  5001
Ntot[8]             235.060  8.088  218.910  234.991  251.035    FALSE 1 1.000  5001
Ntot[9]             242.350  9.801  222.797  242.348  261.853    FALSE 1 1.000  5001
Ntot[10]            249.889 11.908  226.322  249.854  273.437    FALSE 1 1.000  4472
deviance           2357.011  3.900 2351.593 2356.370 2366.672    FALSE 1 1.000  5001
```

The parameter estimates look satisfactory: the demographic rates and the recapture probability correspond well to the data generating values. The estimated observation error of the state-space model is a bit larger than the value used for the data generation. Although the two stage classes could not be distinguished when the population was counted, estimates of stage-specific population sizes are obtained from the integrated population model. The ability to estimate population structure is an important asset of integrated population models, in particular for retrospective and prospective population analyses (see chapters XY). The comparison between true population sizes, counts and population estimates shows that the estimates are usually closer to the truth than the counts (Fig. 5.1). Thus, the integrated population model was successful to get rid of the variability in the counts induced by the random sampling. We also note that the estimated population trajectory follows a straight line, while the true population size fluctuates slightly. The reason for this discrepancy is that the population model assumes absence of demographic stochasticity, while the data generation is performed with demographic stochasticity.

```
u <- col2rgb("grey92")
col.pol <- rgb(u[1], u[2], u[3], alpha = 75, maxColorValue = 255)
par(cex = 1.2)
plot(m3$mean$Ntot, type = "n", ylim = range(c(m3$q2.5$Ntot, m3$q97.5$Ntot)),
ylab = "Population size", xlab = "Year", las = 1, cex = 1.5)
points(m3$mean$Ntot, type = "b", col = "black", pch = 16, lty = 1, cex = 1.5)
points(count, type = "b", col = "cornflowerblue", pch = 1, lty = 2, cex = 1.5)
points(ind$Nu["Total",], type = "b", col = "red", pch = 1, lty = 2, cex = 1.5)
polygon(c(1:T, T:1), c(m3$q2.5$Ntot, m3$q97.5$Ntot[T:1]), border = NA, col =
col.pol)
legend("topleft", legend = c("Truth", "Counts", "Estimates"), pch = c(1, 1,
16), col = c("red", "cornflowerblue", "black"), lty = c(2, 2, 1), bty = "n")
```

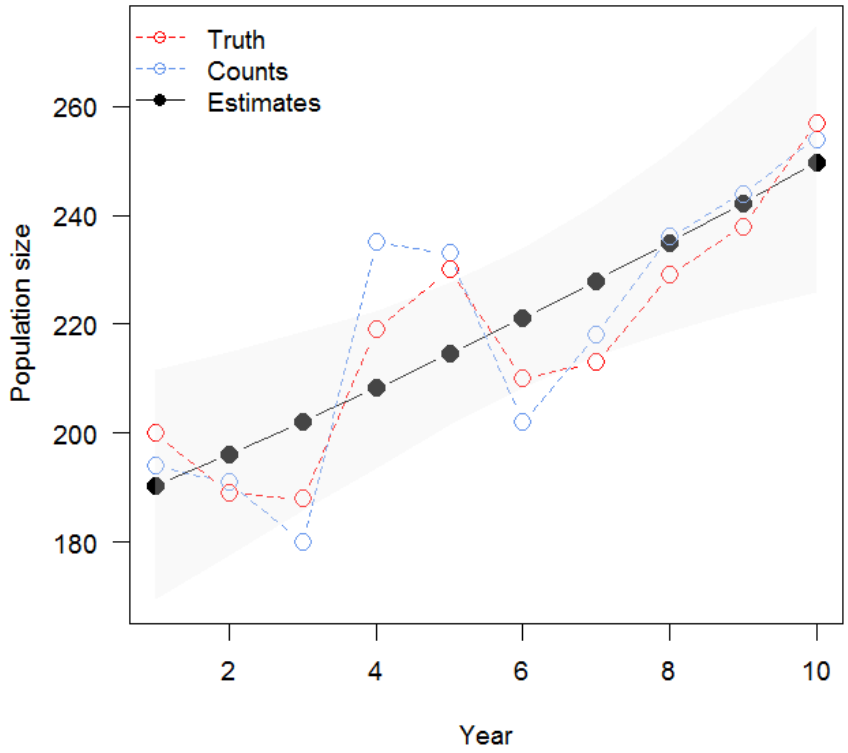


Figure 5.1 True total population size, population counts and population estimates from the simple integrated population model.

This is now the first fully integrated population model that we have developed in this book! Although there are a lot of technical details that one has to master to fit an integrated population model, these models are conceptually not that complicated, aren't they? By the progression of models that we have learnt in chapters 3 and 4, we have seen that integrated population models are a combination of matrix projection models and demographic data models.

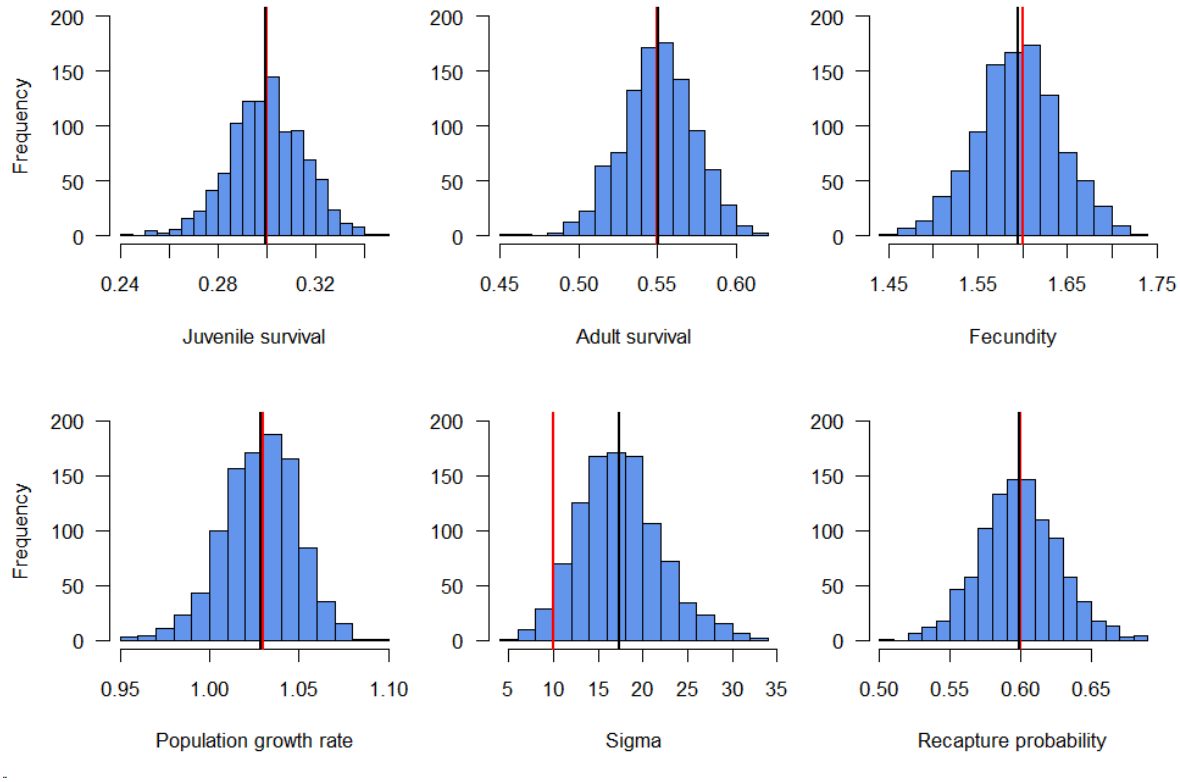


Figure 5.2 Frequency distributions of posterior means of parameters from the integrated population model obtained from 1000 simulations. The red vertical lines indicate the data generation values, the black vertical line the mean of the posterior means. [file: Simulation simple IPM chapter 6.3.txt; R-file: IPM Simulation chapter 6.3 A.Rdata]

The results of the current integrated population model are highly satisfactory - the demographic rates are well estimated and correspond to what we have expected. However, to assess whether our first integrated population model really produces unbiased parameter estimates a small simulation study is necessary. We simulated data sets as above and analysed them with our integrated population model 1000 times and computed histograms of the posterior means of parameters of interest. All demographic parameters, the population growth rate and the recapture probability appear to be unbiased (Fig. 5.2), but the observation error (sigma) is positively biased. Since sigma is the residual error of the integrated population model it does not contain only pure observation errors of the counts, but also lack of fit. As we have discussed already above, our model assumes absence of demographic stochasticity while the data are simulated under demographic stochasticity, thus some lack of fit is expected, which is added to the residual error. To experience that this discussion is really true, you can solve exercise 20.6.2.

The development of the integrated population model in BUGS was not such a great deal, provided that we know the demographic data models and the population projection models. Yet we may ask ourselves, what have we done statistically, how does this integrated analysis work? To understand this better we develop the likelihood of the applied integrated population model (at least symbolically).

The formulation of the likelihood of an integrated population model requires technically that the likelihood of each demographic model is developed first (single data likelihoods). Under

the assumption of independence, the joint likelihood of the integrated population model is the product of the single data likelihoods (Besbeas et al. 2002, Brooks et al. 2004, Béliveau et al. 2016). The assumption of independence is crucial: stricktly speaking it is only fulfilled if the different data sets are sampled from different populations, which is rarely fulfilled in practice. If the data sets are not completely independent in this sense, the likelihood is not the true joint likelihood, but rather a composite likelihood (Béliveau et al. 2016). We will see later what are the consequences of the violatation of the independence assumption and also provide possible solutions in case if the violation of the assumption poses a problem for inference. But for now, we assume that the data sets are independent. We develop the single data likelihood here symbolically. The capture-recapture data (***m***, the m-array) contains information about survival and recapture probabilities and likelihood of the Cormack-Jolly-Seber model is symbolized as $L_{CJS}\left(s_j, s_a, p \middle| \boldsymbol{m}\right)$. The productivity data (***J***) contain information about fecundity is the likelihood of the Poisson regression model is symbolized as $L_P\left(f \middle| \boldsymbol{J}\right)$. Finally, the count data (***y***) contain information about all demographic parameters and the observation (residual) error. The state-space model is composed of three bits, the likelihoods for the initial population size ($L_I\left(\boldsymbol{N}_1\right)$), for the process model ($L_S\left(\boldsymbol{N}, s_j, s_a, f\right)$) and for the observation model ($L_O\left(\boldsymbol{N}, \sigma \middle| \boldsymbol{y}\right)$). Thus, the likelihood of the complete state-space model is $L_{SS}\left(\boldsymbol{N}, s_j, s_a, f, \sigma \middle| \boldsymbol{y}\right) = L_I\left(\boldsymbol{N}_1\right) \times L_O\left(\boldsymbol{N}, \sigma \middle| \boldsymbol{y}\right) \times L_S\left(\boldsymbol{N}, s_j, s_a, f\right)$. Under the assumption of independence the likelihood of the integrated population model is $L_{IPM}\left(\boldsymbol{N}, s_j, s_a, p, f, \sigma \middle| \boldsymbol{m}, \boldsymbol{J}, \boldsymbol{y}\right) = L_I\left(\boldsymbol{N}_1\right) \times L_O\left(\boldsymbol{N}, \sigma \middle| \boldsymbol{y}\right) \times L_S\left(\boldsymbol{N}, s_j, s_a, f\right) \times L_{CJS}\left(s_j, s_a, p \middle| \boldsymbol{m}\right) \times L_P\left(f \middle| \boldsymbol{J}\right)$.

Inference is based on this joint likelihood. In the frequentist mode of analysis the joint likelihood is maximized, which usually requires filtering algorythms such as the Kalman filter (Besbeas et al. 2002, Besbeas et al. 2003, Besbeas et al. 2005, Besbeas and Freeman 2006, Besbeas and Morgan 2012, Gauthier et al. 2007, Péron et al. 2010, Tavecchia et al. 2009, Valpine 2012). In the Bayesian mode of analysis the joint likelihood is combined with the prior distributions of the parameters to get the posteriors of the quantities of interest.

5.4. Flux of information results in more precise parameter estimates in integrated population models

The symbolic formulation of the joint likelihood as the product of the single data likelihoods makes clear that some parameters appear in different components of the joint likelihood. For example occurs juvenile survival in the likelihood of the capture-recapture model and also in the likelihood of the state-space model. By the combination of likelihoods the information regarding common parameters is pooled and this filters into the estimation of the remaining parameters (Brooks et al. 2004). The combination of the single data likelihoods to form the joint likelihood of the integrated population model can well be illustrated by a directed acyclic graph (DAG, Fig. 5.3). In such a graph known quantities (data) are shown by squares while unknown quantities (estimated parameters) are shown by circles. Arrows between nodes indicate dependences among quantities in the model. The single data models are shown by large boxes, and unknown quantities that are included in such boxes show that they are part of the corresponding model. Quantities that appear in two boxes are informed by two data sets.

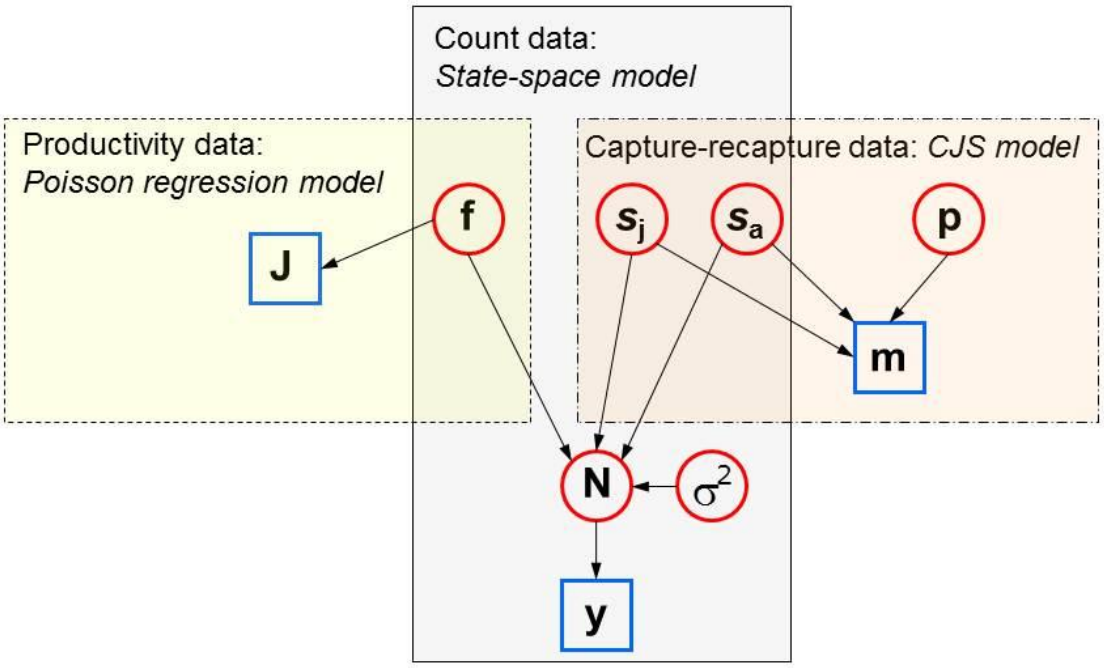


Figure 5.3 Graphical representation of the integrated population model for woodchat shrikes. The graph is similar to a directed acyclic graph (DAG) without the priors. Data are symbolised with small squares, estimated parameters with circles. Large boxes show the individual submodels and the arrows the dependencies among nodes. Circles appearing in two submodels indicate that they are informed by two data sources. For the notation of parameters and data see the text.

Because data sets that are informative about common parameters are jointly analysed in an integrated population model, it is expected that the precision of parameter estimates is improved. Increased precision of these parameter may also filter to other parameters of the complete model, but how the information is flowing is not a priori clear. To experience the flux of information in our integrated population model we conducted a small simulation study (code is provided in appendix XY). Using the same underlying demographic data as before (thus a woodchat shrike) we first created a population from which we sampled the three data sets (capture-recapture data, productivity data, counts). We then analysed these data separately to estimate age-specific survival from the capture-recapture data, fecundity from the productivity data and population growth rate from the state-space model. Next, we jointly analysed the data using the integrated population model. Finally, the length (i.e. number of years) of each data sets was increased in turn three times and the resulting data analysed, again singly and jointly. Three joint analyses have been preformed, each with two original data sets and one longer data set. These simulation steps were repeated 1000 times and we stored the standard deviation of the demographic rates. If the information in the data increases, the standard deviations of the estimates are expected to decline. Thus we expect that the standard deviations of the

parameters to be smaller when they originate from the integrated population models than when they originate from the single data models. Moreover, if the length of a specific data set is increased, we expect that the standard deviation of the demographic rates to which this data set provides direct information (e.g. survival when capture-recapture data set is longer) becomes smaller. If increased precision also filters to other parameters, we expect that the standard deviations of other demographic parameters are also affected positively.

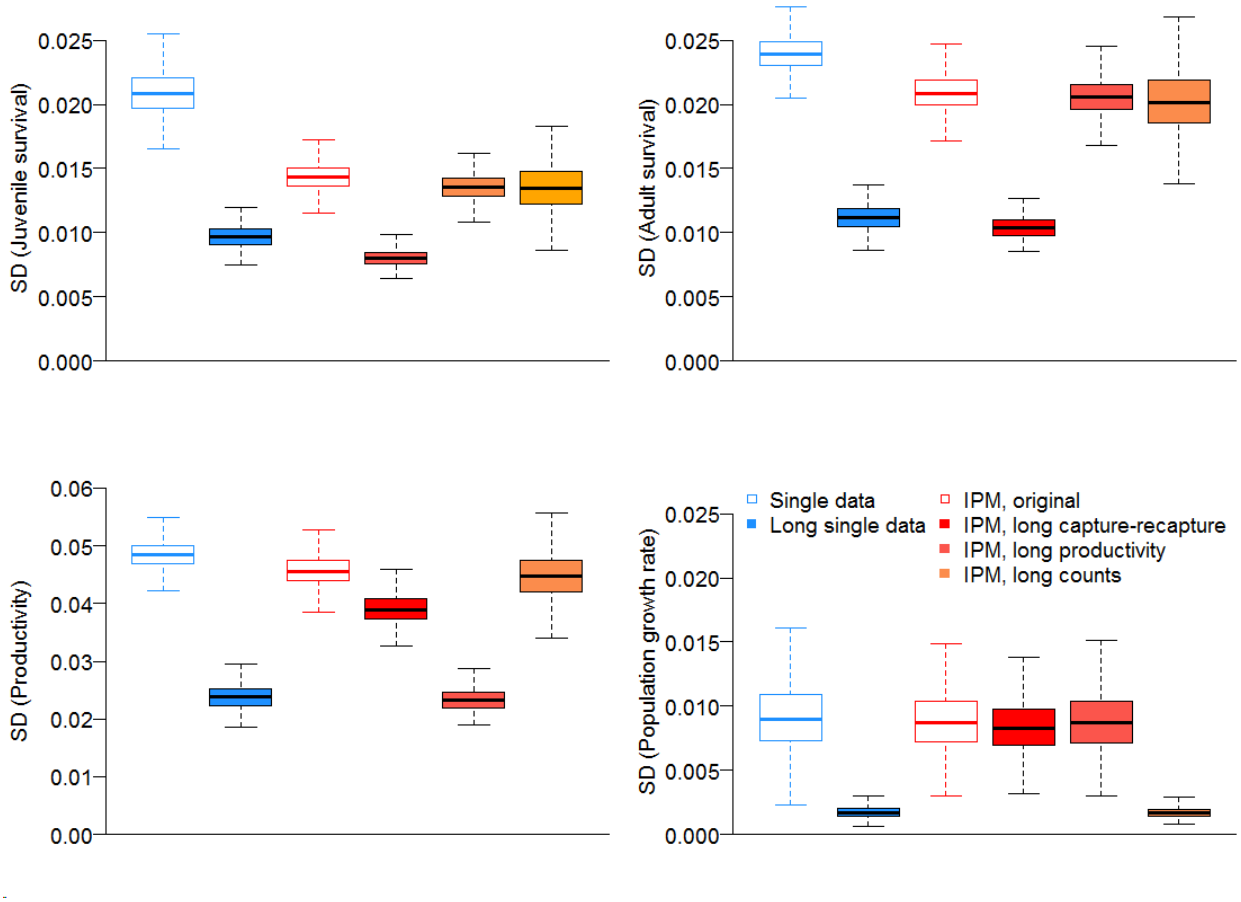


Figure 5.4 Boxplot of the posterior standard deviations of juvenile survival, adult survival, productivity and population growth rate based on 1000 simulations. The blue boxes refer to parameters originating single data models, the red boxes to parameters from integrated population models. The open boxes refer to the original data while the closed boxes refer to data the are 3 times longer.

The simulation study clearly demonstrates that the precision of all demographic parameter is increased when they originate from the integrated population model compared to when they originate from the single data models (Fig. 5.4, compare the open boxplots). The gain in precision is in our example most stricking for juvenile survival – the standard deviation became 31% smaller in the integrated population model compared to the single data model. For adult survival the standard deviation was reduced by 13%, for fecundity by 6% and for the population growth rate by 3%. The reason for the gain in precision when the integrated population model is obvious: the counts contain information about all demographic processes that are operating in the population and this information is exploited with the integrated

population model. Thus, the integrated population model makes more efficient use of the available information.

When the time-series of the data sets was prolonged three times, the standard deviations of all parameters to which the longer data sets provides direct information became significantly smaller, usually by more than 50% (Fig. 5.4, compare open and closed boxplots with the same colour). This is expected and not exciting. However, the increased precision of parameters may also affect the precision of other parameters in integrated population models. This is evident here when the longer time series of capture-recapture data was used. The standard deviation of fecundity got reduced considerably by 14%, while the standard deviation of the population growth rate was marginally reduced only (0.5%). We may wonder how is it possible that fecundity got more precise when a longer time-series of capture-recapture data is used, as this data clearly does not contain any direct information about fecundity. The key is that the count data contain information about fecundity (via the number of recruits which is a function of fecundity and juvenile survival) that is more efficiently exploited when juvenile survival becomes more precise. When the time-series of the other data sets were enlarged, it had almost no effect on the standard deviations of parameters to which the data sets provides no direct information. For example, when the time-series of counts got longer, the standard deviation of the demographic rates did hardly change. This means that the demographic data contributed by far most of the information about the demographic parameters, while the counts hardly contributed. These findings are not general, but likely different when other models or data sets are used.

Increased precision is one advantage of an intergrated population model over conventional separate analyses. Higher precision means that the power to detect impacting factors is higher and that the precision of predicted future population sizes increases. Due to error accumulation over time, even slight increases in precision can make up quite a difference.

5.5. Estimation of demographic parameters without explicit data

Another, and in many practical applications **the** main advantage of integrated population models, is their ability to estimate demographic data for which no specific data have been collected. This is possible, because the count data contain information about all demographic processes and this information can be extracted by the application of an integrated population model. To demonstrate how this works, let's assume that we have not collected any productivity data in our woodchat shrike study. We try to fit an integrated population model to the capture-recapture and the count data. The changes in the code (m3) are trivially easy, only the model for the productivity data has to removed. The prior for fecundity stays the same, i.e. it remains vague. From the data that are thrown to JAGS the productivity data have to be removed and the model run.

```
# Specify the model in BUGS language
cat(file = "m4.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

for (t in 1:(n.occasions-1)){
   sj[t] <- mean.sj
```

```
   sa[t] <- mean.sa
   p[t] <- mean.p
   }

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# State-space model for count data
# Model for the initial population size
N[1,1] ~ dunif(1, 300)
N[2,1] ~ dunif(1, 300)

# Process model over time
for (t in 1:(n.occasions-1)){
   N[1,t+1] <- mean.f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t] + N[2,t], tau.obs)
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]   # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
   pr.a[t,t] <- sa[t]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j])*prod(q[t:(j-1)])*p[j]
      pr.a[t,j] <- prod(sa[t:j])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Derived parameters
# Annual population growth rate
for (t in 1:(n.occasions-1)){
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
# Total population size
for (t in 1:n.occasions){
   Ntot[t] <- N[1,t] + N[2,t]
   }
}
")
```

```
# Bundle data
bugs.data <- list(marr.j = marray[,,1], marr.a = marray[,,2], n.occasions = T,
rel.j = rowSums(marray[,,1]), rel.a = rowSums(marray[,,2]), count = count)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 0.5), mean.sa = runif(1, 0,
1))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.p", "mean.f", "N", "sigma.obs",
"ann.growth.rate", "Ntot")

# MCMC settings
ni <- 12000; nt <- 6; nb <- 2000; nc <- 3

# Call JAGS from R (jagsUI)
m4 <- jags(bugs.data, inits, parameters, "m4.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m4, 3)

JAGS output for model 'm4.jags', generated by jagsUI.
Estimates based on 3 chains of 12000 iterations,
burn-in = 2000 iterations and thin rate = 6,
yielding 5001 total samples from the joint posterior.
MCMC ran for 0.093 minutes at time 2016-06-20 11:36:53.

                     mean      sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj             0.297   0.020   0.259   0.298   0.336    FALSE 1 1.002   985
mean.sa             0.550   0.022   0.507   0.550   0.592    FALSE 1 1.001  5001
mean.p              0.665   0.029   0.607   0.665   0.720    FALSE 1 1.001  1685
mean.f              1.622   0.134   1.374   1.617   1.903    FALSE 1 1.003  1027
N[1,1]             88.320  55.542   4.852  81.539 186.987    FALSE 1 1.005  3703
N[2,1]            102.313  55.672   6.245 107.665 190.316    FALSE 1 1.005  2837
N[1,2]             91.469   5.533  80.821  91.461 102.972    FALSE 1 1.000  5001
N[2,2]            104.888   7.489  91.210 104.697 120.773    FALSE 1 1.001  5001
N[1,3]             94.237   5.336  83.876  94.129 105.187    FALSE 1 1.000  5001
N[2,3]            108.036   6.955  95.203 107.898 122.727    FALSE 1 1.001  5001
N[1,4]             97.100   5.283  86.834  96.981 107.801    FALSE 1 1.000  5001
N[2,4]            111.290   6.490  99.417 111.134 124.774    FALSE 1 1.001  5001
N[1,5]            100.060   5.407  89.788  99.956 110.901    FALSE 1 1.000  5001
N[2,5]            114.654   6.139 103.179 114.572 127.221    FALSE 1 1.000  5001
N[1,6]            103.120   5.728  91.873 103.014 114.331    FALSE 1 1.000  5001
N[2,6]            118.132   5.955 106.984 118.013 130.158    FALSE 1 1.000  5001
N[1,7]            106.286   6.249  93.937 106.310 118.638    FALSE 1 1.000  5001
N[2,7]            121.728   5.990 110.166 121.632 133.662    FALSE 1 1.000  5001
N[1,8]            109.559   6.956  95.760 109.512 123.274    FALSE 1 1.000  5001
N[2,8]            125.446   6.282 113.346 125.324 138.024    FALSE 1 1.000  5001
N[1,9]            112.946   7.835  97.747 112.808 128.842    FALSE 1 1.000  5001
N[2,9]            129.291   6.841 115.762 129.223 143.271    FALSE 1 1.000  5001
N[1,10]           116.449   8.867  99.234 116.265 134.142    FALSE 1 1.000  5001
N[2,10]           133.268   7.652 118.182 133.306 149.056    FALSE 1 1.000  5001
sigma.obs          19.356   5.818  11.396  18.285  34.445    FALSE 1 1.002  1825
ann.growth.rate[1]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[2]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[3]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[4]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[5]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[6]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[7]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[8]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
ann.growth.rate[9]  1.031   0.011   1.009   1.031   1.051    FALSE 1 1.000  5001
Ntot[1]           190.633  11.046 169.753 190.347 213.790    FALSE 1 1.000  5001
Ntot[2]           196.357   9.681 177.986 196.204 216.418    FALSE 1 1.000  5001
Ntot[3]           202.274   8.375 186.274 202.169 219.549    FALSE 1 1.000  5001
Ntot[4]           208.390   7.238 194.529 208.290 223.316    FALSE 1 1.000  5001
```

```
Ntot[5]           214.714  6.462 202.127 214.734 227.746    FALSE 1 1.000  5001
Ntot[6]           221.252  6.303 208.581 221.288 233.784    FALSE 1 1.000  5001
Ntot[7]           228.013  6.926 214.033 228.157 241.566    FALSE 1 1.000  5001
Ntot[8]           235.005  8.272 217.980 235.179 251.001    FALSE 1 1.000  5001
Ntot[9]           242.237 10.162 221.382 242.488 261.821    FALSE 1 1.000  5001
Ntot[10]          249.717 12.452 224.338 249.985 274.171    FALSE 1 1.000  5001
deviance          305.601  3.984 300.036 304.856 315.160    FALSE 1 1.000  5001
```

Mean productivity is very close to the estimate that we obtained from the integrated population model including all three data sets, but the standard deviation is about three time larger than before. The decrease of precision is to be expected, since information about productivity now only originates from the count data. That the precision of demographic parameters for which no data have been collected is generally low, is a common and not unexpected result (Abadi et al. 2010b, Schaub and Fletcher 2015).

When the integrated model allows the estimation of parameters for which no demographic data are collected, it is of interest knowing how far can we go. What about when we only have sampled count and productivity data, but no capture-recapture data? Or if we only have count data? Does this work? Well, we can simply try it out! In the model codes we again have to delete the bits where the likelihood of the specific data is written. First a model is developed that analyses productivity and count data only.

```
# Specify the model in BUGS language
cat(file = "m5.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# State-space model for count data
# Model for the initial population size
N[1,1] ~ dunif(1, 300)
N[2,1] ~ dunif(1, 300)

# Process model over time
for (t in 1:(n.occasions-1)){
   N[1,t+1] <- mean.f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t] + N[2,t], tau.obs)
   }

# Poisson regression model for productivity data
for (i in 1:n.J){
   J[i] ~ dpois(mean.f)
   }
# Derived parameters
# Annual population growth rate
for (t in 1:(n.occasions-1)){
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
# Total population size
```

```
for (t in 1:n.occasions){
   Ntot[t] <- N[1,t] + N[2,t]
   }
}
")

# Bundle data
bugs.data <- list(n.occasions = T, J = P$rep.ind[,1], n.J = nrow(P$rep.ind),
count = count)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0.25, 0.35), mean.sa = runif(1,
0.5, 0.6))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.f", "N", "sigma.obs",
"ann.growth.rate", "Ntot")

# MCMC settings
ni <- 200000; nt <- 90; nb <- 50000; nc <- 3

# Call JAGS from R (jagsUI)
m5 <- jags(bugs.data, inits, parameters, "m5.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m5, 3)


JAGS output for model 'm5.jags', generated by jagsUI.
Estimates based on 3 chains of 2e+05 iterations,
burn-in = 50000 iterations and thin rate = 90,
yielding 5001 total samples from the joint posterior.
MCMC ran for 2.455 minutes at time 2016-06-20 11:40:37.

                      mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
mean.sj              0.337  0.178    0.037    0.334    0.626    FALSE 1 1.021   101
mean.sa              0.488  0.286    0.023    0.490    0.971    FALSE 1 1.021   101
mean.f               1.613  0.049    1.517    1.613    1.713    FALSE 1 1.000  5001
N[1,1]              95.393 53.994    5.652   95.612  187.625    FALSE 1 1.003   686
N[2,1]              95.070 54.022    5.365   94.259  187.515    FALSE 1 1.003   632
N[1,2]             103.283 54.718   11.559  102.756  194.258    FALSE 1 1.020   103
N[2,2]              92.903 54.706    4.279   93.257  187.169    FALSE 1 1.021   102
N[1,3]             106.402 56.279   11.906  105.882  199.343    FALSE 1 1.021   103
N[2,3]              95.697 56.256    4.428   96.084  192.140    FALSE 1 1.021   101
N[1,4]             109.626 57.915   12.396  109.085  204.771    FALSE 1 1.021   102
N[2,4]              98.585 57.879    4.572   99.182  197.827    FALSE 1 1.021   101
N[1,5]             112.959 59.631   12.762  112.711  210.740    FALSE 1 1.021   101
N[2,5]             101.570 59.576    4.738  102.028  203.222    FALSE 1 1.021   101
N[1,6]             116.404 61.429   13.114  115.702  217.042    FALSE 1 1.021   101
N[2,6]             104.656 61.354    4.909  105.246  208.895    FALSE 1 1.021   101
N[1,7]             119.966 63.315   13.554  119.093  223.801    FALSE 1 1.021   100
N[2,7]             107.846 63.214    5.099  108.632  215.179    FALSE 1 1.021   101
N[1,8]             123.648 65.294   13.988  122.599  231.314    FALSE 1 1.021   100
N[2,8]             111.145 65.163    5.265  111.822  221.892    FALSE 1 1.021   101
N[1,9]             127.456 67.369   14.370  126.216  238.602    FALSE 1 1.021   100
N[2,9]             114.555 67.203    5.441  115.167  229.272    FALSE 1 1.021   102
N[1,10]            131.394 69.546   14.708  130.519  246.733    FALSE 1 1.021   100
N[2,10]            118.082 69.341    5.631  118.523  236.561    FALSE 1 1.021   102
sigma.obs           19.407  5.968   11.417   18.167   34.484    FALSE 1 1.001  5001
ann.growth.rate[1]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[2]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[3]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[4]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[5]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[6]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[7]   1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
```

```
ann.growth.rate[8]    1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
ann.growth.rate[9]    1.031  0.010    1.009    1.031    1.051    FALSE 1 1.000  5001
Ntot[1]             190.463 10.946  168.906  190.324  213.031    FALSE 1 1.000  5001
Ntot[2]             196.185  9.652  177.069  196.182  215.533    FALSE 1 1.000  5001
Ntot[3]             202.099  8.418  185.032  201.982  218.857    FALSE 1 1.000  5001
Ntot[4]             208.211  7.348  193.059  208.199  222.849    FALSE 1 1.000  5001
Ntot[5]             214.529  6.613  200.764  214.559  227.740    FALSE 1 1.000  4417
Ntot[6]             221.060  6.440  207.728  221.121  233.881    FALSE 1 1.001  3424
Ntot[7]             227.812  6.985  213.266  227.872  241.622    FALSE 1 1.001  3207
Ntot[8]             234.793  8.214  217.572  234.789  251.143    FALSE 1 1.001  3498
Ntot[9]             242.012  9.977  220.667  242.094  261.969    FALSE 1 1.001  4077
Ntot[10]            249.477 12.138  223.815  249.517  274.120    FALSE 1 1.000  4802
deviance           2138.321  3.487 2133.947 2137.486 2147.015    FALSE 1 1.000  5001
```

It is already now obvious that the estimates of fecundity and population growth rate look according to the results from the previous models, while estimates of survival from both age classes are different. Before we compare the estimates of the different model graphically, we go one step further and construct a model that analyses count data only. This model is not an integrated model anymore, because it uses just a single data source. In fact the applied model is a state-space model, whose process model is written in terms of demographic rates.

```
# Specify the model in BUGS language
cat(file = "m6.jags", "
model {
# Priors and constraints
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# State-space model for count data
# Model for the initial population size
N[1,1] ~ dunif(1, 300)
N[2,1] ~ dunif(1, 300)

# Process model over time
for (t in 1:(n.occasions-1)){
   N[1,t+1] <- mean.f * mean.sj * (N[1,t] + N[2,t])
   N[2,t+1] <- mean.sa * (N[1,t] + N[2,t])
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t] + N[2,t], tau.obs)
   }

# Derived parameters
# Annual population growth rate
for (t in 1:(n.occasions-1)){
   ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t])
   }
# Total population size
for (t in 1:n.occasions){
   Ntot[t] <- N[1,t] + N[2,t]
   }
}
")

# Bundle data
```

```
bugs.data <- list(n.occasions = length(count), count = count)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0.25, 0.35), mean.sa = runif(1,
0.5, 0.6))}

# Parameters monitored
parameters <- c("mean.sj", "mean.sa", "mean.f", "N", "sigma.obs",
"ann.growth.rate", "Ntot")

# MCMC settings
ni <- 1000000; nt <- 570; nb <- 50000; nc <- 3

# Call JAGS from R (jagsUI)
m6 <- jags(bugs.data, inits, parameters, "m6.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)


print(m6, 3)

JAGS output for model 'm6.jags', generated by jagsUI.
Estimates based on 3 chains of 1e+06 iterations,
burn-in = 50000 iterations and thin rate = 570,
yielding 5001 total samples from the joint posterior.
MCMC ran for 1.656 minutes at time 2016-06-20 11:44:49.

                      mean     sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj              0.299  0.265   0.014   0.190   0.918    FALSE 1 1.034    77
mean.sa              0.597  0.284   0.038   0.645   0.987    FALSE 1 1.003   574
mean.f               2.944  2.602   0.146   1.990   9.117    FALSE 1 1.019   141
N[1,1]              95.679 54.883   5.609  95.458 188.064    FALSE 1 1.000  4648
N[2,1]              95.052 54.854   6.076  94.083 187.712    FALSE 1 1.000  5001
N[1,2]              82.534 54.338   8.018  73.228 189.823    FALSE 1 1.003   619
N[2,2]             113.868 54.742   7.384 122.079 195.204    FALSE 1 1.003   565
N[1,3]              85.019 55.905   8.259  75.458 194.821    FALSE 1 1.003   611
N[2,3]             117.245 56.204   7.498 126.001 199.714    FALSE 1 1.003   565
N[1,4]              87.587 57.544   8.530  77.793 200.919    FALSE 1 1.003   603
N[2,4]             120.735 57.742   7.719 130.019 203.768    FALSE 1 1.003   567
N[1,5]              90.242 59.258   8.807  80.150 206.480    FALSE 1 1.003   597
N[2,5]             124.341 59.362   7.966 134.131 208.710    FALSE 1 1.003   569
N[1,6]              92.988 61.052   9.083  82.708 212.670    FALSE 1 1.003   590
N[2,6]             128.068 61.066   8.241 138.759 214.257    FALSE 1 1.003   571
N[1,7]              95.826 62.929   9.260  85.452 219.144    FALSE 1 1.003   585
N[2,7]             131.920 62.861   8.453 142.821 219.651    FALSE 1 1.003   574
N[1,8]              98.762 64.893   9.551  88.117 226.496    FALSE 1 1.003   579
N[2,8]             135.901 64.751   8.671 147.171 226.362    FALSE 1 1.003   577
N[1,9]             101.798 66.948   9.830  91.004 233.787    FALSE 1 1.003   575
N[2,9]             140.017 66.742   8.897 151.913 234.215    FALSE 1 1.003   582
N[1,10]            104.938 69.100  10.098  93.740 241.257    FALSE 1 1.003   570
N[2,10]            144.273 68.837   9.180 156.256 242.038    FALSE 1 1.003   586
sigma.obs           19.310  5.868  11.438  18.083  34.493    FALSE 1 1.000  5001
ann.growth.rate[1]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[2]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[3]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[4]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[5]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[6]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[7]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[8]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
ann.growth.rate[9]   1.030  0.010   1.009   1.030   1.050    FALSE 1 1.000  5001
Ntot[1]            190.731 11.138 169.476 190.194 214.173    FALSE 1 1.000  4367
Ntot[2]            196.403  9.788 177.781 195.998 217.181    FALSE 1 1.000  4386
Ntot[3]            202.264  8.503 186.185 201.961 220.080    FALSE 1 1.001  4573
Ntot[4]            208.322  7.393 194.174 208.157 223.934    FALSE 1 1.001  5001
Ntot[5]            214.583  6.643 201.732 214.495 228.595    FALSE 1 1.001  5001
Ntot[6]            221.055  6.493 208.484 220.992 234.253    FALSE 1 1.001  5001
Ntot[7]            227.746  7.099 213.844 227.697 242.307    FALSE 1 1.001  5001
Ntot[8]            234.663  8.410 217.702 234.607 251.311    FALSE 1 1.000  5001
```

```
Ntot[9]           241.815 10.260 221.619 241.804 261.973    FALSE 1 1.000  5001
Ntot[10]          249.211 12.509 224.226 249.267 273.754    FALSE 1 1.000  5001
deviance           85.761  3.201  82.032  84.948  93.913    FALSE 1 1.001  4450
```

To get converge in the state-space model long MCMC chains are necessary. The estimated population growth rate looks much alike the estimates from the integrated model, but the demographic parameters do not that look good. In order to visualize the results of all models, we plot the posterior distributions of the demographic rates and compare them among models.

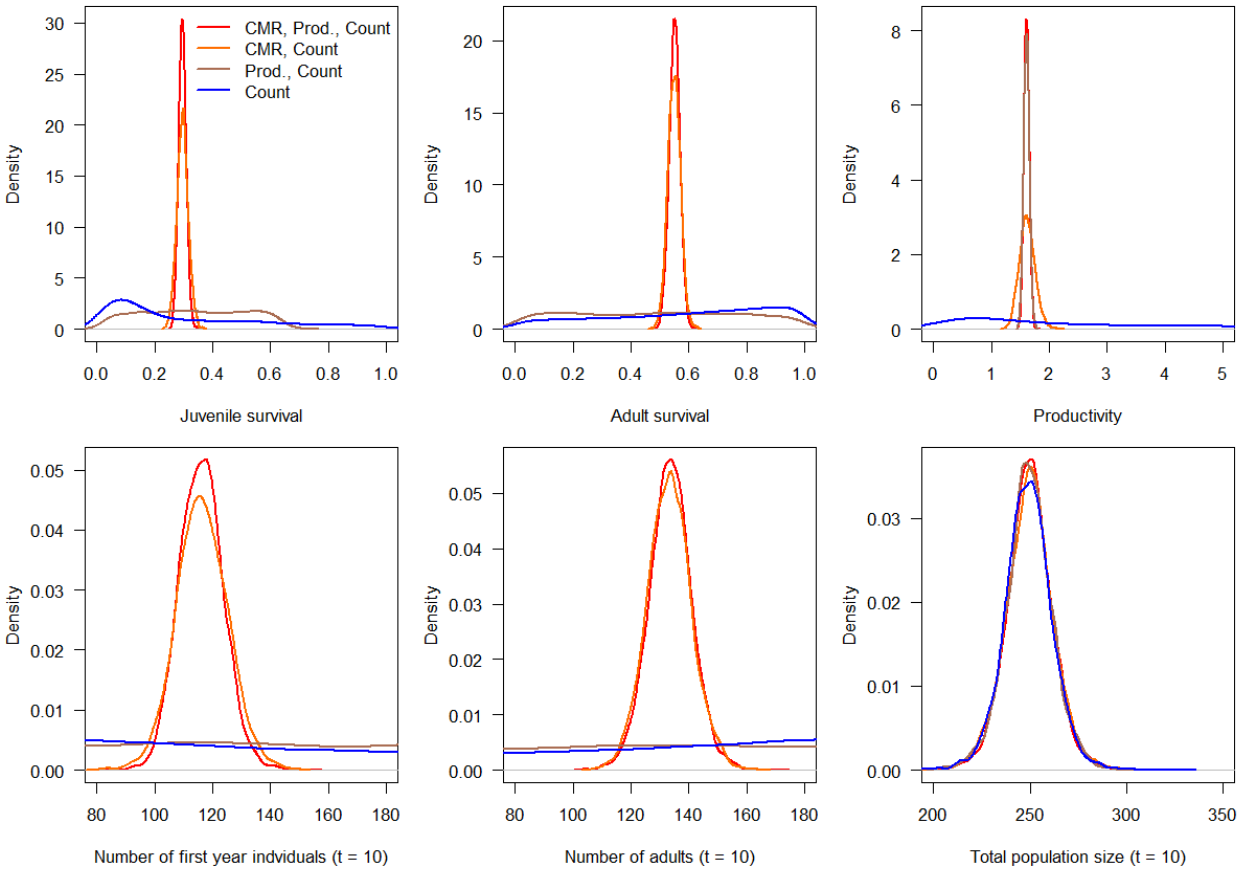


Figure 5.5 Posterior distributions of juvenile and adult survival, fecundity as well as the number of first year individuals, adults and total population site at last occasion from four different models that use different data sets.

Survival of both age classes and population size are well estimated with the integrated population models that either use all available data or capture-recapture and count data (Fig. 5.5). Productivity is well estimated when either all, capture-recapture and count, or productivity and count data are used. Yet, in the latter set-up, survival and stage-specific population size are not well estimated. The estimate of productivity is also acceptable (clear peak of the posterior, hardly biased) when counts and capture-recapture data are combined. When only counts are analysed, none of the demographic rates can be estimated. The posterior distributions are essentially flat. The estimation of the stage-specific population sizes works well when at least

count and capture-recapture data are available, while the estimation of the total population size is possible in all four situations. This means that if only counts are available, we get correct estimates of total population size, but no demographic parameters nor stage-specific population sizes can be estimated. Moreover, if the total population size is correctly estimated, also the growth rate is correct. Yet, the decomposition of total population size into stage-specific population sizes or of the population growth rate into demographic contributions requires correct estimates of demographic rates and thus more data than just counts.

```
co <- colorRampPalette(c("red", "orange", "blue"))(4)
par(mfrow = c(2,3), las = 1, mar = c(4,4,1,1), cex = 1.05)
plot(density(m3$sims.list$mean.sj), xlim = c(0, 1), main = "", xlab =
"Juvenile survival", lwd = 2, col = co[1])
lines(density(m4$sims.list$mean.sj), col = co[2], lwd = 2)
lines(density(m5$sims.list$mean.sj), col = co[3], lwd = 2)
lines(density(m6$sims.list$mean.sj), col = co[4], lwd = 2)
legend("topright", legend = c("CMR, Prod., Count", "CMR, Count", "Prod.,
Count", "Count"), col = co, lwd = rep(2,4), bty = "n")
plot(density(m3$sims.list$mean.sa), xlim = c(0, 1), main = "", xlab = "Adult
survival", lwd = 2, col = co[1])
lines(density(m4$sims.list$mean.sa), col = co[2], lwd = 2)
lines(density(m5$sims.list$mean.sa), col = co[3], lwd = 2)
lines(density(m6$sims.list$mean.sa), col = co[4], lwd = 2)
plot(density(m3$sims.list$mean.f), xlim = c(0, 5), main = "", xlab =
"Productivity", lwd = 2, col = co[1])
lines(density(m4$sims.list$mean.f), col = co[2], lwd = 2)
lines(density(m5$sims.list$mean.f), col = co[3], lwd = 2)
lines(density(m6$sims.list$mean.f), col = co[4], lwd = 2)
plot(density(m3$sims.list$N[,1,10]), xlim = c(80, 180), main = "", xlab =
"Number of first year indviduals (t = 10)", lwd = 2, col = co[1])
lines(density(m4$sims.list$N[,1,10]), col = co[2], lwd = 2)
lines(density(m5$sims.list$N[,1,10]), col = co[3], lwd = 2)
lines(density(m6$sims.list$N[,1,10]), col = co[4], lwd = 2)
plot(density(m3$sims.list$N[,2,10]), xlim = c(80, 180), main = "", xlab =
"Number of adults (t = 10)", lwd = 2, col = co[1])
lines(density(m4$sims.list$N[,2,10]), col = co[2], lwd = 2)
lines(density(m5$sims.list$N[,2,10]), col = co[3], lwd = 2)
lines(density(m6$sims.list$N[,2,10]), col = co[4], lwd = 2)
plot(density(m3$sims.list$Ntot[,10]), xlim = c(200, 350), main = "", xlab =
"Total population size (t = 10)", lwd = 2, col = co[1])
lines(density(m4$sims.list$Ntot[,10]), col = co[2], lwd = 2)
lines(density(m5$sims.list$Ntot[,10]), col = co[3], lwd = 2)
lines(density(m6$sims.list$Ntot[,10]), col = co[4], lwd = 2)
```

5.6. Example with a more complex life history

To illustrate further the construction of an integrated population model, we now show a model for a red kite (*Milvus milvus*) population. This species has a more complex life cycle than the woodchat shrike, the single data models become more complicated and we aim to include demographic stochasticity in the model. Despite the increased complexity remain the conceptual steps to construct the integrated population model remain the same as before: 1) formulate a model that links demographic with stage-specific population sizes (matrix projection model), 2) write the likelihood of the each data set, and 3) construct the joint likelihood for making inference. While constructing and running the model for the red kite we can emphasise solutions to challenges that appear typical in more complex intergrated population models such

as the selection of starting values or the specification of priors for the stage-specific population sizes in the first year. We also show the inclusion of demographic stochasticity and its consequences.

The red kite (Fig. XY) is endemic to Europe and of conservation concern in some European countries. It is long-lived and most individuals start to reproduce at the age of 3 years, some already with 2 years. We assume that we have collected data from a large study area. Each year the number of occupied nests is recorded, but there may be errors because some nests were not detected and because some nests were wrongly recorded as occupied, while in reality they were not in the given year. In addition the nestlings of some nests are marked with rings and wing flags, the latter allow resightings from distance. When an individual was resighted it was recorded whether it is an experienced breeder (meaning that it is breeding in the actual year or has been observed breeding previously) or has not started to reproduce yet. Finally, the number of fledglings is collected from another sample of nests. The study is run for 12 years.

The data the we will use are simulated in a similar way as shown in chapter 6.3. The demographic parameter used to simulate the data are $s_1 = 0.45$, $s_2 = 0.68$ and $s_3 = 0.83$ (age-specific survival), $\alpha = 0.3$ (probability to start reproduction at age of 2 years) and $f = 1.66$ (number of feldglings per reproducing female). We load the data

```
setwd("O:\\IPM Book")
load ("RedKite.Rdata")
```

As a first step for the construction of any integrated population model, a model that links the demographic rates with stage-specific population sizes has to be defined. Such a model is, of course, a population projection model. We construct a females-based model with a pre-breeding census. The youngest individuals in this model are 1 (or nearly so) year old at the time just before reproduction starts. These individuals do not reproduce yet. Because two years old individuals may or may not reproduce, there are two classes (types) of two year old individuals: individuals that are first-time breeder and individuals that do not breed yet. From the third year of age all individuals are assumed to reproduce, and the demographic parameters do not change anymore. The life cycle graph for the red kite population is shown in Fig. 6.7.

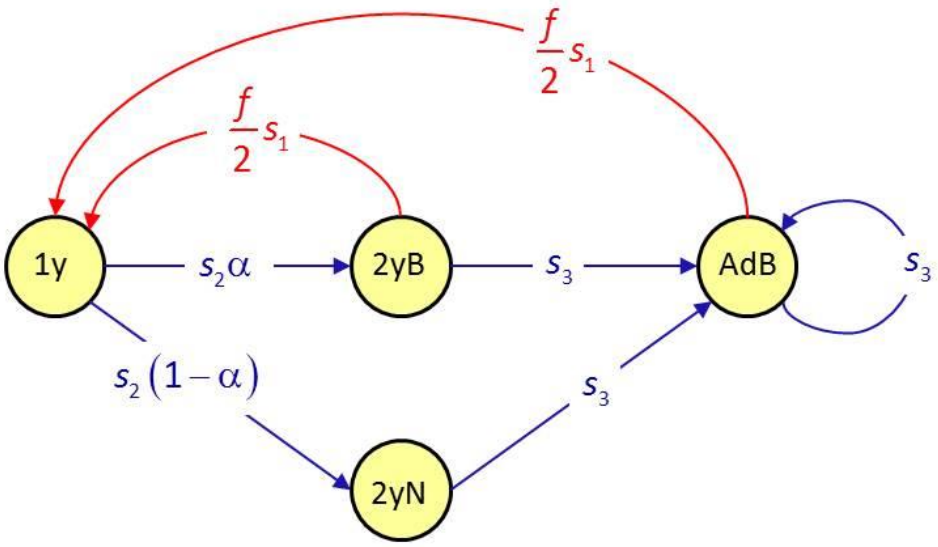


Figure 5.7 Life cycle graph for the red kite with a pre-breeding census. The nodes refer to 1-year old individuals (1y), 2-year old non-breeders (2yN), 2-year old first time breeders (2yB) and to at least 3-year old individuals which are all breeding (AdB). The survival process is shown in blue, the recruitment process in red. The associated parameters are age-specific survival (s_1, s_2, s_3), the probability to start reproduction at age 2 (α) and fecundity (f).

From the life cycle graph we can write the projection matrix model:

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix}_{t+1} = \begin{bmatrix} 0 & 0 & \frac{f_t}{2}s_{1,t} & \frac{f_t}{2}s_{1,t} \\ s_{2,t}(1-\alpha_t) & 0 & 0 & 0 \\ s_{2,t}\alpha_t & 0 & 0 & 0 \\ 0 & s_{3,t} & s_{3,t} & s_{3,t} \end{bmatrix}_t \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix}_t$$

Note that the N_1 refers to the 1-year old individuals (1y in Fig. 5.7), N_2 to 2yN, N_3 to 2yB and N_4 to AdB. For the integrated population model we want to include demographic stochasticity and therefore use stochastic distributions to write the population model. The most appropriate distributions are the Poisson for the recruitment process and the Binomial for the survival processes, and thus we have,

$$N_{1,t+1} \sim \text{Poisson}\left(N_{3,t} \frac{f_t}{2} s_{1,t} + N_{4,t} \frac{f_t}{2} s_{1,t} \right)$$

$$N_{2,t+1} \sim \text{Binomial}\left(N_{1,t}, s_{2,t} \left(1-\alpha_t\right) \right)$$

$$N_{3,t+1} \sim \text{Binomial}\left(N_{1,t}, s_{2,t} \alpha_t \right)$$

$$N_{4,t+1} \sim \text{Binomial}\left(N_{2,t} + N_{3,t} + N_{4,t}, s_{3,t} \right)$$

These equations correspond exactly to the projection model written in matrix form above, with the only difference that demographic stochasticity is included.

As a second step to construct the integrated population model, we now have to develop the likelihoods of the data sets that we have sampled. We will fit these single data models to the data in order to see whether they are properly working. Such a careful stepwise approach for the development of a complex integrated population model is generally adviseable, because errors are more easy to spot. If we construct the integrated population model in one step and it does not run, we may have to step back to the single data model anyway for debugging. We start with the state-space model that we will apply to the count data. The state process equations we have already defined with the model above, this model decribes the true (or what we think it is true) development of the four stages over time. There are many options for the observation model, which links the true states with our observations (counts). Here we assume that the counts are correct on average, but that errors in both directions (missing occupied nests and counting nests that are actually not occupied) can occur. It is reasonable to assume in addition that the errors are larger when population size is larger. Therefore a log-Normal model seems a good choice for the observation model. Our sampling protocol is restricted to breeding individuals, thus the expected count is the sum of $N_{3,t}$ and $N_{4,t}$. The observation model is therefore

$$\log\left(C_t\right) \sim \text{Normal}\left(\log\left(N_{3,t} + N_{4,t} \right), \sigma^2 \right)$$

where C_t is the count in year t and σ^2 is the residual variance. The code for this state-space model is given below, it is not much different to the codes that we have seen in earlier places of the book.

```
# Specify the model in BUGS language
cat(file = "m5.jags", "
model {
# -------------------------------------------------
# Parameters:
# s: stage-specific survival probabilities
# alpha: probability of start reproduction at age = 2 years
# -------------------------------------------------
# Stages (S):
# N1: 1-year old individuals
# N2: 2-year old non-breeders
# N3: 2-year old first-time breeders
# N4: adults (at least 3 years old)
# -------------------------------------------------

# 1. Priors and constraints
# 1.1. Demographic parameters
for (t in 1:(n.occasions-1)){
   s1[t] <- mean.s[1]
   s2[t] <- mean.s[2]
```

```
   s3[t] <- mean.s[3]
   alpha[t] <- mean.alpha
   }
for (u in 1:3){
   mean.s[u] ~ dunif(0, 1)     # Priors for mean stage-spec. survival
   }
mean.alpha ~ dunif(0, 1)       # Priors for prob. start reproduction at age 2y
for (t in 1:n.occasions){
   f[t] <- mean.f
   }
mean.f ~ dunif(0, 10)     # Priors for mean productivity

# 1.2. Initial population sizes (discrete uniform)
N[1,1] ~ dcat(pN1)
N[2,1] ~ dcat(pN2)
N[3,1] ~ dcat(pN3)
N[4,1] ~ dcat(pN4)

# 1.3. Residual variance
sigma.obs ~ dunif(0.005, 10)
tau.obs <- pow(sigma.obs, -2)


# 2. Likelihood of the state-space model
# 2.1. State transition process
for (t in 1:(n.occasions-1)){
   N[1,t+1] ~ dpois(f[t] / 2 * s1[t] * (N[3,t] + N[4,t]))
   N[2,t+1] ~ dbin((1 - alpha[t]) * s2[t], N[1,t])
   N[3,t+1] ~ dbin(alpha[t] * s2[t], N[1,t])
   N[4,t+1] ~ dbin(s3[t], (N[2,t] + N[3,t] + N[4,t]))
   }

# 2.2. Observation process
for (t in 1:n.occasions){
   logCount[t] ~ dnorm(log(N[3,t] + N[4,t]), tau.obs)
   }
}
")
```

For the initial population sizes we need to specify priors, and there are many choices for this. The important condition that has to be fulfilled is that the generated number is a positive integer. This is requested by the Binomial distribution in the state process (note that WinBUGS is more liberal and allows also continuous numbers as the Binomial total). To fulfill this condition we may use Possion or negative binomial distributions, log-Normal distributions whose generated numbers are rounded to integers, or Normal distributions truncated to positive values and whose generated numbers are rounded to integers. Here we want to use the discrete uniform distribution. Since this distribution does not exists in JAGS, we implement it by using the categorical distribution. The categorical distribution requires as parameter a vector of length *T* with probabilities that indicate for each value {1…T} how likely it is chosen. These probabilities need to sum to one. The following R function creates the needed vector for a discrete uniform distribution between A and B (where A>0):

```
disc.unif <- function(A, B){
   pprob <- c(rep(0, A-1), rep(1/(B-A+1), (B-A+1)))
   return(pprob)
   }
```

If A = 5 and B = 7, the resulting vector is

```
disc.unif(5,7)
[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.3333333 0.3333333 0.3333333
```

A further difficulty for the specification of the priors of the initial stage-specific population sizes is that they are requested for all defined stages of the state-space model, but we have counts only from individuals of some stages, namely those that are actually breeding. The number of the younger individuals is completely unknown, and hence it is difficult to find appropriate ranges of the priors. Of course we could specify very wide priors of all stage classes, but this may result in difficulties to run the model. Moreover, to start the MCMC starting values for the stage-specific population sizes have to be given and it can be decisive to chose them in a good way otherwise the model may not run. A useful approach to address both challenges is the calculation of the stable stage distribution from a matrix projection model that is alike the state process of the state-space model. We can then scale the counts and have a rough guess about the size of the states that are not observed. Since the demographic rates are unknown, we need to inspect the literature to set up the matrix projection model. For the red kite, we know from literature that survival in the first year is about 0.4, about 0.7 in the second year and 0.8 later. We assume that the probability to start to reproduce at age 2 is 0.5 and productivity is 1.7. The Leslie matrix and the scaled stable state distribution are:

```
A <- matrix(c(0, 0, 1.7/2*0.4, 1.7/2*0.4,
           (1-0.5)*0.7, 0, 0, 0,
           0.5*0.7, 0, 0, 0,
           0, 0.8, 0.8, 0.8), ncol = 4, byrow = TRUE)

u <- which.max(Re(eigen(A)$values))
revec <- Re(eigen(A)$vectors[,u])
stable.stage <- revec/sum(revec)

[1]  0.23151302 0.08015739 0.08015739 0.60817221
```

The count of breeding individuals in the first study year equals 129 – it is the sum of the number of individuals of the third and fourth stage classes. The expected numbers of individuals in the four stages are then

```
129*stable.stage[1]/(stable.stage[3]+stable.stage[4])
[1] 43.38791

129*stable.stage[2]/(stable.stage[3]+stable.stage[4])
[1] 15.02231

129*stable.stage[3]/(stable.stage[3]+stable.stage[4])
[1] 15.02231

129*stable.stage[4]/(stable.stage[3]+stable.stage[4])
[1] 113.9777
```

These number of just rough guidelines, their computation assumed that the demographic rates were correct and that the population has reached a stable stage distribution. To express somewhat diffuse priors we create discrete uniform priors +/- 15 around these values. Thus, we specify the needed probabilities for the discrete uniform priors as

```
pN1 <- disc.unif(28, 58)
pN2 <- disc.unif(1, 30)
pN3 <- disc.unif(1, 30)
pN4 <- disc.unif(99, 129)
```

These probabilities need to be given as data. We bundle all the needed data

```
# Bundle data
bugs.data <- list(logCount = log(Count), n.occasions = length(Count), pN1 =
pN1, pN2 = pN2, pN3 = pN3, pN4 = pN4)

# Initial values
inits <- function(){list(mean.s = c(0.4, 0.6, 0.8), mean.alpha = runif(1, 0,
0.5), mean.f = 1.5)}

# Parameters monitored
parameters <- c("mean.s", "mean.alpha", "mean.f", "N", "sigma.obs")

# MCMC settings
ni <- 20000; nt <- 1; nb <- 5000; nc <- 3

# Call JAGS from R
m5 <- jags(bugs.data, inits, parameters, "m5.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m5, digits = 3)

JAGS output for model 'm5.jags', generated by jagsUI.
Estimates based on 3 chains of 20000 iterations,
burn-in = 5000 iterations and thin rate = 1,
yielding 45000 total samples from the joint posterior.
MCMC ran for 0.511 minutes at time 2016-10-17 14:07:46.

               mean      sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
mean.s[1]     0.426   0.263   0.057   0.380   0.958    FALSE 1.000 1.010  1226
mean.s[2]     0.414   0.262   0.046   0.370   0.948    FALSE 1.000 1.032    70
mean.s[3]     0.837   0.124   0.552   0.865   0.994    FALSE 1.000 1.011   253
mean.alpha    0.530   0.293   0.034   0.548   0.977    FALSE 1.000 1.015   141
mean.f        4.181   2.645   0.401   3.698   9.548    FALSE 1.000 1.027    82
N[1,1]       42.889   8.912  28.000  43.000  58.000    FALSE 1.000 1.000 24805
N[2,1]       14.784   8.503   1.000  14.000  30.000    FALSE 1.000 1.001  2238
N[3,1]       14.920   8.320   1.000  15.000  30.000    FALSE 1.000 1.003   599
N[4,1]      113.558   8.415  99.000 113.000 128.000    FALSE 1.000 1.003   753
…
N[1,12]     138.742 133.184   4.000 103.000 557.000    FALSE 1.000 1.102    47
N[2,12]      17.151  17.950   0.000  11.000  66.000     TRUE 1.000 1.036   131
N[3,12]      23.430  23.539   0.000  15.000  84.000     TRUE 1.000 1.024   153
N[4,12]     161.085  25.160 105.000 164.000 202.000    FALSE 1.000 1.029   848
sigma.obs     0.144   0.044   0.085   0.136   0.254    FALSE 1.000 1.009   437
deviance    -14.324   4.251 -21.036 -14.917  -3.735    FALSE 0.988 1.018   197
```

The model works technically, so this is good. We do not look at the estimates, because most parameter are not separately estimable anyway, as we have seen already (Fig. 5.2).

The next data set that we have sampled are the ringing and subsequent resighting data. Only nestlings are ringed and marked with wing tags. When an individual is resighted it may be 2-years old and non-breeding (state 2), 3-years old and non-breeding (state 3), 3-years old and breeding (first time breeder, state 4) or older than 3 years (state 5). Thus, in the field it is noted, whether an individual is observed as a breeder or as a non-breeder. The current age of each individual is known becaue for each of them we know the birth – and hence the necessary coding for the capture histories can be done. Such data allow the estimation of age-specific

probability to start with reproduction (Pradel et al. 1997, Pradel and Lebreton 1999, Szostek et al. 2014). For the multistate model we will use the multinomial likelihood (section 4.XY) and thus include the 5 states “nestling”, “1-year old”, “2-year old non-breeding”, “2-year old first time breeder”, and “adult”. Thus, the state transition matrix for the multistate model is the following:

$$\begin{bmatrix} 0 & s_{1,t} & 0 & 0 & 0 \\ 0 & 0 & s_{2,t}(1-\alpha_t) & s_{2,t}\alpha_t & 0 \\ 0 & 0 & 0 & 0 & s_{3,t} \\ 0 & 0 & 0 & 0 & s_{3,t} \\ 0 & 0 & 0 & 0 & s_{3,t} \end{bmatrix}$$

The rows in this transition matrix refer to the states in year *t* (in the order given above) and the columns refer to the states in year *t*+1. It is imperative to understand that the states in the multistate model may or may not be identical to the stages of the state-space model, and they must not be confused. The states in the multistate model are defined such that the desired demographic parameters (here survival and probability of first time breeding) can be estimated given the sampled data.

The vector with the reencounter probabilities is the following:

$$\begin{bmatrix} 0 \\ p_{1,t} \\ p_{1,t} \\ p_{2,t} \\ p_{2,t} \end{bmatrix}$$

We allow in the model the probability to resight individuals that have never reproduced (p_1) to be different from the resighting probability of individuals that are either first time or experienced breeders (p_2). Obvioulsy nestlings can never be reencountered as nestlings. It is relatively straightforward to write code for this model:

```
# Specify the model in BUGS language
cat(file = "m6.jags", "
model {
# -------------------------------------------------
# Parameters:
# s: age-specific survival probabilities
# alpha: probability to start reproduction at age = 2 years
# p: recapture probability
# -------------------------------------------------
# States (S):
# 1 nestling
# 2 1-year old non-breeder
# 3 2-year old non-breeder
# 4 2-year old first time breeder
# 5 experienced breeder
# -------------------------------------------------

# 1. Priors and constraints
for (t in 1:(n.occasions-1)){
```

```
   s1[t] <- mean.s[1]
   s2[t] <- mean.s[2]
   s3[t] <- mean.s[3]
   alpha[t] <- mean.alpha
   p1[t] <- mean.p[1]
   p2[t] <- mean.p[2]
   }
for (u in 1:3){
   mean.s[u] ~ dunif(0, 1)     # Priors for mean state-spec. survival
   }
mean.alpha ~ dunif(0, 1)       # Priors for prob. start reproduction at age 2y
for (u in 1:2){
   mean.p[u] ~ dunif(0, 1)     # Priors for mean state-spec. resighting
   }

# 2. Multistate capture-recapture model
# Define state-transition and reencounter probabilities
   for (t in 1:(n.occasions-1)){
      psi[1,t,1] <- 0
      psi[1,t,2] <- s1[t]
      psi[1,t,3] <- 0
      psi[1,t,4] <- 0
      psi[1,t,5] <- 0
      psi[2,t,1] <- 0
      psi[2,t,2] <- 0
      psi[2,t,3] <- s2[t] * (1-alpha[t])
      psi[2,t,4] <- s2[t] * alpha[t]
      psi[2,t,5] <- 0
      psi[3,t,1] <- 0
      psi[3,t,2] <- 0
      psi[3,t,3] <- 0
      psi[3,t,4] <- 0
      psi[3,t,5] <- s3[t]
      psi[4,t,1] <- 0
      psi[4,t,2] <- 0
      psi[4,t,3] <- 0
      psi[4,t,4] <- 0
      psi[4,t,5] <- s3[t]
      psi[5,t,1] <- 0
      psi[5,t,2] <- 0
      psi[5,t,3] <- 0
      psi[5,t,4] <- 0
      psi[5,t,5] <- s3[t]

      po[1,t] <- 0
      po[2,t] <- p1[t]
      po[3,t] <- p1[t]
      po[4,t] <- p2[t]
      po[5,t] <- p2[t]

      # Calculate probability of non-encounter (dq) and reshape the array for
the encounter probabilities
      for (s in 1:ns){
         dp[s,t,s] <- po[s,t]
         dq[s,t,s] <- 1-po[s,t]
         } # s
      for (s in 1:(ns-1)){
         for (m in (s+1):ns){
            dp[s,t,m] <- 0
            dq[s,t,m] <- 0
            } # s
         } # m
      for (s in 2:ns){
```

```
         for (m in 1:(s-1)){
            dp[s,t,m] <- 0
            dq[s,t,m] <- 0
            } # s
         } # m
      } # t

# Define the multinomial likelihood
for (t in 1:((n.occasions-1)*ns)){
   marr[t,1:(n.occasions*ns-(ns-1))] ~ dmulti(pr[t,], rel[t])
   }

# Define the cell probabilities of the m-array
# Define matrix U: product of probabilities of state-transition and non-
encounter
for (t in 1:(n.occasions-2)){
   U[(t-1)*ns+(1:ns), (t-1)*ns+(1:ns)] <- ones
   for (j in (t+1):(n.occasions-1)){
      U[(t-1)*ns+(1:ns), (j-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns), (j-
2)*ns+(1:ns)] %*% psi[,t,] %*% dq[,t,]
      }
   }
   U[(n.occasions-2)*ns+(1:ns), (n.occasions-2)*ns+(1:ns)] <- ones

# Define the cell probabilities of the multistate m-array
for (t in 1:(n.occasions-2)){
   pr[(t-1)*ns+(1:ns),(t-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns),(t-1)*ns+(1:ns)]
%*% psi[,t,] %*% dp[,t,]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr[(t-1)*ns+(1:ns), (j-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns), (j-
1)*ns+(1:ns)] %*% psi[,j,] %*% dp[,j,]
      }
   }
   pr[(n.occasions-2)*ns+(1:ns), (n.occasions-2)*ns+(1:ns)] <-
psi[,n.occasions-1,] %*% dp[,n.occasions-1,]

# Below main diagonal
for (t in 2:(n.occasions-1)){
   for (j in 1:(t-1)){
      pr[(t-1)*ns+(1:ns),(j-1)*ns+(1:ns)] <- zero
      } #j
   } #t

# Last column: probability of non-recapture
for (t in 1:((n.occasions-1)*ns)){
   pr[t,(n.occasions*ns-(ns-1))] <- 1-sum(pr[t,1:((n.occasions-1)*ns)])
   } #t
}
")
```

This model requires multistate m-array data. We create such data from the multistate capture-recapture data and bundle them. Finally, specify the necessary ingredients and run the model.

```
# Bundle data
ms.arr <- marray(CH)
ns <- 5
bugs.data <- list(marr = ms.arr, n.occasions = ncol(CH), rel =
rowSums(ms.arr), ns = ns, zero = matrix(0, ncol = ns, nrow = ns), ones =
diag(ns))

# Initial values
```

```
inits <- function(){list(mean.s = runif(3, 0, 1))}

# Parameters monitored
parameters <- c("mean.s", "mean.alpha", "mean.p")

# MCMC settings
ni <- 1000; nt <- 1; nb <- 500; nc <- 3

# Call JAGS from R
m6 <- jags(bugs.data, inits, parameters, "m6.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m6, digits = 3)

JAGS output for model 'm6.jags', generated by jagsUI.
Estimates based on 3 chains of 1000 iterations,
burn-in = 500 iterations and thin rate = 1,
yielding 1500 total samples from the joint posterior.
MCMC ran for 0.216 minutes at time 2017-08-28 11:52:29.

              mean    sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.s[1]    0.383 0.036   0.320   0.382   0.462    FALSE 1 1.000  1500
mean.s[2]    0.715 0.065   0.585   0.716   0.854    FALSE 1 1.003  1500
mean.s[3]    0.846 0.023   0.801   0.847   0.888    FALSE 1 1.003  1500
mean.alpha   0.236 0.045   0.153   0.234   0.328    FALSE 1 1.001  1194
mean.p[1]    0.319 0.031   0.260   0.318   0.379    FALSE 1 1.001  1500
mean.p[2]    0.702 0.029   0.645   0.703   0.757    FALSE 1 1.003   663
deviance   424.158 3.252 419.618 423.581 431.599    FALSE 1 1.007   315
```

The final data set is the productivity data. These data are summarized annually, that is, from a number of surveyed broods (B_t) we have tallied up the total number of fledglings (J_t). The advantage of analysing aggregated data compared to individual data is that computation time is reduced. On the other side, individual traits such as the age of the mother cannot be modeled. The model to analyse the aggregated productivity data is here a simple Poisson regression model, and the following bit of code does that.

```
# Specify the model in BUGS language
cat(file = "m7.jags", "
model {
# 1. Priors and constraints
for (t in 1:n.occasions){
   f[t] <- mean.f
   }
mean.f ~ dunif(0, 10)         # Priors for mean productivity

# 2. Likelihood
for (t in 1:n.occasions){
   J[t] ~ dpois(B[t] * f[t])
   }
}
")
```

We load the data and fit the model.

```
# Bundle data
bugs.data <- list(J = J, B = B, n.occasions = length(J))

# Initial values
inits <- function(){list(mean.f = runif(1, 0, 2))}
```

```
# Parameters monitored
parameters <- c("mean.f")

# MCMC settings
ni <- 1000; nt <- 1; nb <- 500; nc <- 3

# Call JAGS from R
m7 <- jags(bugs.data, inits, parameters, "m7.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m7, digits = 3)

JAGS output for model 'm7.jags', generated by jagsUI.
Estimates based on 3 chains of 1000 iterations,
burn-in = 500 iterations and thin rate = 1,
yielding 1500 total samples from the joint posterior.
MCMC ran for 0.024 minutes at time 2017-08-28 11:53:29.

           mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
mean.f    1.608 0.043  1.524  1.606  1.693    FALSE 1 1.006   301
deviance 88.543 1.492 87.508 87.973 92.919    FALSE 1 1.009  1500
```

We now have defined the likelihoods (models) for all the three data sets that we have sampled, we fitted these models to the data sets and all of them run. The final step is now the construction of the integrated population model to jointly analyse the three data sets. As we have discussed before this technically means that we have to write a joint likelihood. Owing to the BUGS language this is easy, we just have to bundle the three models into a common one and the integration works by giving the same name to parameters that are shared among data sets. For the priors we have to make sure that they are defined only once. The complete integrated population model thus looks like

```
# Specify the model in BUGS language
cat(file = "m8.jags", "
model {
# -----------------------------------------------
# Parameters:
# s: age-specific survival probabilities
# alpha: probability to start reproduction at age = 2 years
# p: recapture probability
# F: productivity
# sigma.obs: residual error
# -----------------------------------------------
# Stages for the population model:
# N1 1-year old individuals
# N2 2-year old non-breeders
# N3 2-year old first-time breeders
# N4 adults (at least 3 years old)
# -----------------------------------------------
# States for the multistate capture-recapture model:
# 1 nestling
# 2 1-year old non-breeder
# 3 2-year old non-breeder
# 4 2-year old first time breeder
# 5 experienced breeder
# -----------------------------------------------

# 1. Priors and constraints
# 1.1. Demographic parameters
for (t in 1:(n.occasions-1)){
```

```
   s1[t] <- mean.s[1]
   s2[t] <- mean.s[2]
   s3[t] <- mean.s[3]
   alpha[t] <- mean.alpha
   p1[t] <- mean.p[1]
   p2[t] <- mean.p[2]
   }
for (u in 1:3){
   mean.s[u] ~ dunif(0, 1)     # Priors for mean state-spec. survival
   }
mean.alpha ~ dunif(0, 1)       # Priors for prob. start reproduction at age 2y
for (u in 1:2){
   mean.p[u] ~ dunif(0, 1)     # Priors for mean state-spec. resighting
   }
for (t in 1:n.occasions){
   F[t] <- mean.f
   }
mean.f ~ dunif(0, 10)     # Priors for mean productivity

# 1.2. Initial population sizes (discrete uniform)
N[1,1] ~ dcat(pN1)
N[2,1] ~ dcat(pN2)
N[3,1] ~ dcat(pN3)
N[4,1] ~ dcat(pN4)

# 1.3. Residual variance
sigma.obs ~ dunif(0.005, 10)
tau.obs <- pow(sigma.obs, -2)

# 2. State-space model for count data
# 2.1. State transition process
for (t in 1:(n.occasions-1)){
   N[1,t+1] ~ dpois(F[t] / 2 * s1[t] * (N[3,t] + N[4,t]))
   N[2,t+1] ~ dbin(s2[t] * (1-alpha[t]), N[1,t])
   N[3,t+1] ~ dbin(s2[t] * alpha[t], N[1,t])
   N[4,t+1] ~ dbin(s3[t], (N[2,t] + N[3,t] + N[4,t]))
   }

# 2.2. Observation matrix
for (t in 1:n.occasions){
   logCount[t] ~ dnorm(log(N[3,t] + N[4,t]), tau.obs)
   }

# 3. Multistate capture-recapture model for multistate capture-recapture data
# 3.1. Define state-transition and reencounter probabilities
   for (t in 1:(n.occasions-1)){
      psi[1,t,1] <- 0
      psi[1,t,2] <- s1[t]
      psi[1,t,3] <- 0
      psi[1,t,4] <- 0
      psi[1,t,5] <- 0
      psi[2,t,1] <- 0
      psi[2,t,2] <- 0
      psi[2,t,3] <- s2[t] * (1-alpha[t])
      psi[2,t,4] <- s2[t] * alpha[t]
      psi[2,t,5] <- 0
      psi[3,t,1] <- 0
      psi[3,t,2] <- 0
      psi[3,t,3] <- 0
      psi[3,t,4] <- 0
      psi[3,t,5] <- s3[t]
      psi[4,t,1] <- 0
      psi[4,t,2] <- 0
```

```
      psi[4,t,3] <- 0
      psi[4,t,4] <- 0
      psi[4,t,5] <- s3[t]
      psi[5,t,1] <- 0
      psi[5,t,2] <- 0
      psi[5,t,3] <- 0
      psi[5,t,4] <- 0
      psi[5,t,5] <- s3[t]

      po[1,t] <- 0
      po[2,t] <- p1[t]
      po[3,t] <- p1[t]
      po[4,t] <- p2[t]
      po[5,t] <- p2[t]

      # Calculate probability of non-encounter (dq) and reshape the array for
the encounter probabilities
      for (s in 1:ns){
         dp[s,t,s] <- po[s,t]
         dq[s,t,s] <- 1-po[s,t]
         } # s
      for (s in 1:(ns-1)){
         for (m in (s+1):ns){
            dp[s,t,m] <- 0
            dq[s,t,m] <- 0
            } # s
         } # m
      for (s in 2:ns){
         for (m in 1:(s-1)){
            dp[s,t,m] <- 0
            dq[s,t,m] <- 0
            } # s
         } # m
      } # t

# Define the multinomial likelihood
for (t in 1:((n.occasions-1)*ns)){
   marr[t,1:(n.occasions*ns-(ns-1))] ~ dmulti(pr[t,], rel[t])
   }

# Define the cell probabilities of the m-array
# Define matrix U: product of probabilities of state-transition and non-
encounter (this is just done because there is no product function for matrix
multiplication in JAGS)
for (t in 1:(n.occasions-2)){
   U[(t-1)*ns+(1:ns), (t-1)*ns+(1:ns)] <- ones
   for (j in (t+1):(n.occasions-1)){
      U[(t-1)*ns+(1:ns), (j-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns), (j-
2)*ns+(1:ns)] %*% psi[,t,] %*% dq[,t,]
      }
   }
   U[(n.occasions-2)*ns+(1:ns), (n.occasions-2)*ns+(1:ns)] <- ones

# Define the cell probabilities of the multistate m-array
for (t in 1:(n.occasions-2)){
   pr[(t-1)*ns+(1:ns),(t-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns),(t-1)*ns+(1:ns)]
%*% psi[,t,] %*% dp[,t,]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr[(t-1)*ns+(1:ns), (j-1)*ns+(1:ns)] <- U[(t-1)*ns+(1:ns), (j-
1)*ns+(1:ns)] %*% psi[,j,] %*% dp[,j,]
      }
   }
```

```
   pr[(n.occasions-2)*ns+(1:ns), (n.occasions-2)*ns+(1:ns)] <-
psi[,n.occasions-1,] %*% dp[,n.occasions-1,]

# Below main diagonal
for (t in 2:(n.occasions-1)){
   for (j in 1:(t-1)){
      pr[(t-1)*ns+(1:ns),(j-1)*ns+(1:ns)] <- zero
      } #j
   } #t

# Last column: probability of non-recapture
for (t in 1:((n.occasions-1)*ns)){
   pr[t,(n.occasions*ns-(ns-1))] <- 1-sum(pr[t,1:((n.occasions-1)*ns)])
   } #t

# 4. Poisson regression model for productivity data
# 4.1. Likelihood
for (t in 1:n.occasions){
   J[t] ~ dpois(B[t] * F[t])
   }
}
")
```

The data that JAGS need are of course the three data sets, as well as the paramaters to specify the discrete uniform priors for the stage-specific population sizes in the first year.

```
# Bundle data
ns <- 5
bugs.data <- list(marr = ms.arr, n.occasions = ncol(CH), rel =
rowSums(ms.arr), ns = ns, zero = matrix(0, ncol = ns, nrow = ns), ones =
diag(ns), J = J, B = B, logCount = log(Count), pN1 = pN1, pN2 = pN2, pN3 =
pN3, pN4 = pN4)
```

Finally, the list of initial values is composed of the inits from the single data models. In addition, we also specify initial values for the stage-specific population sizes. This is not always needed, but sometimes an integrated population model only starts running if starting values for them are given. Since stage-specific population sizes are stochastic, starting values could be given for all of them. Yet, here we only want to give them for the first year. Stage-specific population sizes are defined as a matrix in the model, hence the starting values must also be specified by a matrix of the same dimensions.

```
Ninit <- matrix(NA, nrow = 4, ncol = length(Count))
Ninit[1,1] <- 43
Ninit[2,1] <- 15
Ninit[3,1] <- 15
Ninit[4,1] <- 114
```

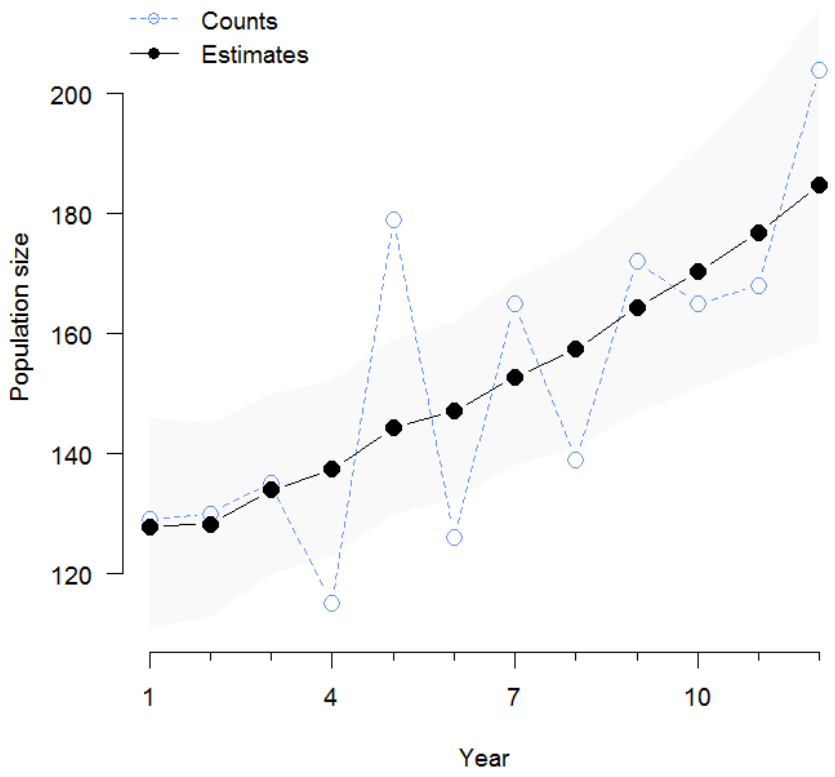
A good choice for the starting values are the stage-specific population sizes that we have calculated before based on the stable stage distribution. The other elements of the matrix are NA, indicating that no starting values are chosen by us (but JAGS does it).

```
# Initial values
inits <- function(){list(mean.s = runif(3, 0, 1), N = Ninit)}

# Parameters monitored
parameters <- c("mean.s", "mean.alpha", "mean.p", "mean.f", "N", "sigma.obs")
```

```
# MCMC settings
ni <- 5000; nt <- 1; nb <- 2500; nc <- 3

# Call JAGS from R
m8 <- jags(bugs.data, inits, parameters, "m8.jags", n.chains = nc, n.thin =
nt, n.iter = ni, n.burnin = nb)

print(m8, digits = 3)


JAGS output for model 'm8.jags', generated by jagsUI.
Estimates based on 3 chains of 5000 iterations,
burn-in = 2500 iterations and thin rate = 1,
yielding 7500 total samples from the joint posterior.
MCMC ran for 1.22 minutes at time 2017-08-28 11:56:35.

              mean     sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.s[1]    0.387  0.035   0.322   0.386   0.458    FALSE 1 1.009   247
mean.s[2]    0.713  0.065   0.588   0.712   0.836    FALSE 1 1.006   344
mean.s[3]    0.850  0.016   0.817   0.851   0.879    FALSE 1 1.007   323
mean.alpha   0.231  0.048   0.142   0.229   0.331    FALSE 1 1.006   566
mean.p[1]    0.318  0.033   0.258   0.317   0.385    FALSE 1 1.003   691
mean.p[2]    0.702  0.029   0.644   0.702   0.757    FALSE 1 1.001  1718
mean.f       1.608  0.042   1.526   1.608   1.690    FALSE 1 1.001  1466
N[1,1]      41.781  8.762  28.000  41.000  57.000    FALSE 1 1.002   783
N[2,1]      15.271  8.367   1.000  15.000  30.000    FALSE 1 1.003   740
N[3,1]      14.889  8.139   1.000  15.000  29.000    FALSE 1 1.002  1280
N[4,1]     112.914  8.301  99.000 112.000 128.000    FALSE 1 1.002  1206
N[1,2]      38.698  7.358  25.000  39.000  54.000    FALSE 1 1.005   401
N[2,2]      22.603  6.123  12.000  22.000  35.000    FALSE 1 1.001  7500
N[3,2]       6.812  3.074   2.000   7.000  13.000    FALSE 1 1.004   548
N[4,2]     121.438  8.512 105.000 121.000 139.000    FALSE 1 1.006   346
N[1,3]      40.720  7.527  27.000  40.000  57.000    FALSE 1 1.003   910
N[2,3]      20.820  5.021  12.000  21.000  31.000    FALSE 1 1.005   484
N[3,3]       6.167  2.816   1.000   6.000  12.000    FALSE 1 1.001  5178
N[4,3]     127.723  7.783 113.000 128.000 143.000    FALSE 1 1.000  7500
N[1,4]      40.967  7.428  27.000  41.000  56.000    FALSE 1 1.004   633
N[2,4]      22.706  5.218  13.000  22.000  34.000    FALSE 1 1.006   455
N[3,4]       6.497  2.856   2.000   6.000  13.000    FALSE 1 1.001  7391
N[4,4]     130.852  7.275 117.000 131.000 145.000    FALSE 1 1.003   840
N[1,5]      42.664  7.428  29.000  42.000  58.000    FALSE 1 1.008   260
N[2,5]      21.904  5.194  12.000  22.000  33.000    FALSE 1 1.004  1006
N[3,5]       7.041  2.957   2.000   7.000  13.000    FALSE 1 1.001  2056
N[4,5]     137.180  7.596 123.000 137.000 152.000    FALSE 1 1.002  1113
N[1,6]      44.679  7.798  31.000  44.000  61.000    FALSE 1 1.004   493
N[2,6]      23.401  5.168  14.000  23.000  34.000    FALSE 1 1.007   400
N[3,6]       6.781  2.896   2.000   7.000  13.000    FALSE 1 1.001  3256
N[4,6]     140.283  7.773 125.000 140.000 156.000    FALSE 1 1.006   667
N[1,7]      46.223  8.144  31.000  46.000  63.000    FALSE 1 1.013   167
N[2,7]      24.224  5.495  14.000  24.000  36.000    FALSE 1 1.002  4687
N[3,7]       7.373  3.100   2.000   7.000  14.000    FALSE 1 1.004   558
N[4,7]     145.362  8.187 130.000 145.000 162.000    FALSE 1 1.011   309
N[1,8]      48.036  8.280  33.000  48.000  65.000    FALSE 1 1.007   296
N[2,8]      25.497  5.761  15.000  25.000  38.000    FALSE 1 1.010   259
N[3,8]       7.434  3.107   2.000   7.000  14.000    FALSE 1 1.001  3122
N[4,8]     149.990  8.652 133.000 150.000 167.000    FALSE 1 1.020   198
N[1,9]      49.361  8.576  34.000  49.000  67.000    FALSE 1 1.004   601
N[2,9]      26.262  5.696  16.000  26.000  38.000    FALSE 1 1.003   757
N[3,9]       8.093  3.316   2.000   8.000  15.000    FALSE 1 1.001  3013
N[4,9]     156.266  9.022 139.000 156.000 174.000    FALSE 1 1.020   129
N[1,10]     52.067  8.911  35.000  52.000  71.000    FALSE 1 1.015   138
N[2,10]     27.033  5.985  16.000  27.000  40.000    FALSE 1 1.000  7500
N[3,10]      8.071  3.281   2.000   8.000  15.000    FALSE 1 1.003   863
```

```
N[4,10]    162.298 10.147 143.000 162.000 183.000    FALSE 1 1.019   157
N[1,11]     53.212  9.522  36.000  53.000  73.000    FALSE 1 1.004   483
N[2,11]     28.833  6.287  17.000  29.000  42.000    FALSE 1 1.012   217
N[3,11]      8.586  3.428   3.000   8.000  16.000    FALSE 1 1.003   854
N[4,11]    168.233 11.496 146.000 168.000 191.525    FALSE 1 1.019   223
N[1,12]     55.133  9.991  37.000  55.000  76.000    FALSE 1 1.005   414
N[2,12]     29.033  6.596  17.000  29.000  43.000    FALSE 1 1.001  1938
N[3,12]      8.951  3.564   3.000   9.000  16.000    FALSE 1 1.001  2161
N[4,12]    175.774 13.400 151.000 175.000 204.000    FALSE 1 1.019   161
sigma.obs    0.137  0.036   0.086   0.132   0.224    FALSE 1 1.006   411
deviance   496.967  4.581 489.727 496.422 507.432    FALSE 1 1.004   694
```

```
N.breeding <- m8$sims.list$N[,3,] + m8$sims.list$N[,4,]
lower <- upper <- pred <- numeric()
n.years <- length(Count)
for (t in 1:n.years){
   lower[t] <- quantile(N.breeding[,t], 0.025)
   upper[t] <- quantile(N.breeding[,t], 0.975)
   pred[t] <- mean(N.breeding[,t])
   }
m1 <- min(c(Count, lower))
m2 <- max(c(Count, upper))
u <- col2rgb("grey92")
col.pol <- rgb(u[1], u[2], u[3], alpha = 75, maxColorValue = 255)
par(cex = 1.2)
plot(pred, type = "n", ylim = c(m1, m2), ylab = "Population size", xlab =
"Year", las = 1, cex = 1.5, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(1, n.years, 3), labels = seq(1, n.years, 3))
axis(1, at = 1:n.years, labels = rep("", n.years), tcl = -0.25)
polygon(c(1:n.years, n.years:1), c(lower, upper[n.years:1]), border = NA, col
= col.pol)
points(Count, type = "b", col = "cornflowerblue", pch = 1, lty = 2, cex = 1.5)
points(pred, type = "b", pch = 16, lty = 1, cex = 1.5)
legend("topleft", legend = c("Counts", "Estimates"), pch = c(1, 16), col =
c("cornflowerblue", "black"), lty = c(2, 1), bty = "n")
```


Counts
Estimates
200
180
160
140
120
Population size
1
4
7
10
Year

10. Population viability analysis

10.1. Introduction

Smart population management requires the definition of an objective which is ideally formulated in quantitative terms and knowledge about how a population is likely to develop in the future. The goal in the context of conservation is to prevent a population of a desirable species from reaching size zero or else to help it reach some minimal size at which it may be deemed sustainable. We therefore have to know whether the focal population is in danger at all and whether any actions are necessary. A popular measure of "danger" is the *extinction probability* of a population, which is the conceptual foundation of the red list criteria of the IUCN (Mace et al. 2008). If a population is classified as endangered, we may want to know which conservation action is best to improve its status most efficiently or at the least cost. Such an assessment requires prediction of the future development of a population under different management options and the estimation of extinction probability. The most efficient management option is identified by the greatest reduction of the extinction probability. The objective in the context of harvesting may be the achievement of a maximum sustainabe removal. A look into the future behavior of a population is therefore necessary to determine how many individuals can be removed from the population in a sustainable way, i.e. that the long-term survival of the population is not at risk. For a pest species, the goal may be extinction or a low acceptable population density which again requires an understanding of how the size of the population is likely to change when some action is taken. In all cases we need a clear understanding of how a population is likely to evolve in the future either under the same conditions as currently experienced or under consideration of demographic, environmental or other changes. To achieve this goal, a technique known as *population viability analysis (PVA)* has been developed in the 1980s in the context of conservation (Shaffer 1981, Gilpin and Soulé 1986). In two different fields, wildlife management and fisheries, similar population analyses employed to predict sustainable harvest have had an even longer history (Lebreton 2005). In this chaper, we focus on population viability analysis, but we emphasize the close relationship with models for sustainable harvesting and fisheries stock assessments, since they all require the knowledge about population demography to predict the future likely development of a population.

There is not a unique definition of a population viability analysis. Morris and Doak (2002) defined a PVA as a *"quantitative method to predict the likely future status of a population"*, and Reed et al. (2002) as *"an assessment of risk that a population reaches some threshold in some time"*. The latter definition is more specific and fouses primarily on extinction probability and related quantities while the former is broader and includes any method to assess the future status of a population. Predicting the future development of a population is an important element in both definitions. The reasons for population extinctions can be broadly classified into two groups: first, extinction may result from the steady decline of a population due to the change of environmental conditions that occurred in the past and is still persisting. This has been called the *declining population paradigm* in the seminal paper of Caughley (1994). Unless environmental conditions improve, extinction is certain, the only uncertainty being when exactly the extinction is going to happen. To help such a population we need to know the reason of the decline and therefore have to investigate the past development of the population. Retrospective population analyses (chapter XY) play an important role in this context. Predicting the future development of a population is then a means to tell whether an improvement of the

environmental conditions through some suggested management action is already enough to prevent extinction or to reduce its risk to an acceptable level.

Second, extinction may result solely from chance events, e.g. due to demographic stochasticity, environmental catastrophes or loss of genetic diversity, even if the mean demographic rates are high enough to ensure a population growth rate larger than 1. Naturally, such chance events have greater impacts the smaller a population is (Gilpin and Soulé 1986), hence, this school of thought has been called the *small population paradigm* (Caughley 1994). In contrast to a declining population, extinction of such a small population is not certain and therefore we want to estimate its likelihood, for which population models again play a major role. If we find that extinction is likely, then we would typically want to identify the most efficient management actions to reduce that extinction risk.

Population viability analysis in a narrower sense deals with extinction probability and related quantities (Reed et al. 2002) and therefore requires population projection into the future. Projection must be based on a model that includes the major genetic and demographic features of the population. Genetic aspects can be important because small populations lose genetic heterozygosity due to inbreeding depression and genetic drift (Allendorf and Ryman 2002, Lande 1988), which in turn can affect the demography of the individuals (Keller 1998) or reduce their evolutionary ability to adapt to altered environmental conditions in the future. Knowledge about the demographic parameters such as the mean and the variability of demographic rates or the mechanisms and magnitude of density-dependence is all needed for the construction of a population projection model. Most population viability analyses conducted in the past have excluded genetic aspects, most of all because knowledge about how the loss of genetic heterozygosity affects demographic rates is rarely available (Reed et al. 2002) and to a lesser degree because genetis is likely to be less important than demography, in particular in the short term (Lande 1988, Boyce 1992). Here we also deal with demographic PVA only, but we would not want to claim that genetic aspects for population viability are always unimportant. Another class of demographic PVA is based on count data only (Dennis et al. 1991, Morris and Doak 2002) and we don't focus on such count-based PVAs either. Instead, we illustrate the classical PVA which is based on some sort of age- or stage-structured, stochastic population model, which is the focus of most IPMs and of much of this book.

In this chapter we illustrate the use of an integrated population model to predict the likely future trajectory of a study population (or multiple populations). We will see that a PVA is really a very natural and relatively minor extension of any IPM. We discuss some challenges of demographic PVAs and show how Bayesian IPM can help to cope with them. We do not provide a full review of PVA here - for that, see Boyce (1992), Beissinger and Westphal (1998), Beissinger (2002), and Reed et al. (2002).

10.2. Challenges for demographic population viability analyses

Demographic population viability analyses deal with the future development of a population. The major challenge is therefore that we want to make predictions about a stochastic system based on imperfect knowledge about that system and there are different ways in which uncertainty afflicts such predictions, including structural, or process, uncertainty and parameter uncertainty.

The development of the population is affected by stochastic events, such as demographic and environmental stochasticity, resulting in *process uncertainty*. If we can describe stochasticity by some rules (typically by statistical distributions), we can make statements about the expected behavior of a population, but this is unavoidably accompanied

by uncertainty. Typically it has to be assumed that the distributions of the demographic rates remain stationary, which means that they remain the same also in the future. If the stationarity assumption is unlikely, process uncertainty greatly increase. Indeed, for an unstationary system, prediction into the future may become impossible. We never have perfect knowledge about how a population works - there is always more or less uncertainty. For example, the demographic rates are not known perfectly, but must be estimated from data, or we do not know on which demographic rate density-dependence operates and what is the precise functional form of density-dependence. Uncertainty about parameters can result also from observer bias or from differential opinions among experts about parameters for which no data are available. Such *parameter uncertainty* affects the accuracy of population predictions in addition to the process uncertainty. It is important to include both sources of uncertainty to correctly express the current state of knowledge about the population and to get honest predictions.

Soon after their invention PVA received severe criticism for several reasons (reviewed in Morris and Doak 2002). Here we briefly review a few important points and briefly mention how the PVA that we will develop in this chapter can help to overcome these problems.

First, it has been critizised that PVA are sometimes performed for species for which adequate data are lacking. It is often a matter of fact that for species that are rare and have small populations detailed demographic information is lacking, and thus estimates of demographic rates are highly uncertain and may be biased. Ignoring parameter uncertainty is not recommended, otherwise the outcome of a PVA becomes too optimistic in the sense that we claim to be more certain than we actually are (McGowan et al. 2011). Standard practice is the inclusion of parameter uncertainty, but predictions of future population sizes can then be associated with such a high level of uncertainty that inference becomes practically meaningless (Ludwig 1999). In IPMs, we exploit the available information in a more efficient way than do typical PVAs, hence parameters estimates become more precise (chapter XY) and arguably also more robust. Therefore, PVAs based on IPMs may be beneficial when data are scarce.

Second, it has been argued that estimates of temporal variability of the demographic rates may be contaminated by sampling variability which results in overestimation of process variability unless properly accounted for and in too high extinction risk in a PVA. Fortunately, models with temporal random effects (see Section xx—xx) can correct estimates of temporal variability for sampling variability (Link 1999). The temporal variation of demographic rates may also be correlated, and such covariances have to be included in PVA (Fieberg and Ellner 2001) to avoid bias in the extinction probability, which may be positive or negative depending on the precise correlation structure of the demographic rates. The multivariate Normal distribution can be used to estimate such correlations (Link and Barker 2005). In IPM it is even possible to include process correlation of parameters that were estimated from different data sets or for which no explicit data were sampled (Schaub et al. 2013). Thus, with IPMs we can again make a step forward.

Third, PVA have been criticized because density-dependence has often been ignored. Constructing models without density-dependence may be justified in some cases when population size is low, but not in general. Density-dependence usually dampens population fluctuations and can greatly reduce the probability of extinction (Ginzburg et al. 1990). Even the form of density-dependence can affect extinction probability (Mills et al. 1996, Middleton and Nisbet 1997). In IPMs, the inclusion of density-dependence is straightforward in principle (Abadi et al. 2012), though the identification of the functional form of density-dependence remains a challenge.

Fourth, spatial heterogeneity and dispersal can stabilize population fluctuations but can also result in more complex dynamics (Beissinger and Westphal 1998). Hence, inclusion of the spatial structure of a population or set of populations is a further important aspect of PVA that has often been neglected. Whether or not spatial structure can be included depends to a large extent on the quality of the available data. IPMs have been developed for spatially structured populations (McCrea et al. 2010, Schaub et al. 2015, Péron et al. 2010, Borysiewicz et al. 2009).

Fifth, PVA results have sometimes been presented without associated measures of uncertainty. Clearly, PVAs must account for process stochasticity and parameter uncertainty, both of which engender uncertainty about future population size and other summaries of the population. This uncertainty must be presented – a single value of a future population size without a measure of uncertainty is meaningless (Beissinger and Westphal 1998, Reed et al. 2002, Ludwig 1996b). However, as outlined above, inferences from PVAs that include process and parameter uncertainty are often very imprecise, and the uncertainty increases the further we project a population into the future (Fieberg and Ellner 2000).

Finally, a sixth point of critique was therefore that the results of a PVA that honestly include all kinds of process stochasticity and uncertainty and parameter uncertainty may become meaningless because they are too uncertain (Ludwig 1999). Inferences can be improved if the population is not projected too far into the future and if instead of absolute, relative extinction risks are compared for different management options. Such relative risks are often more meaningful than are the single values of absolute risk under a certain management action because process stochasticity and parameter uncertainty may cancel out to some degree when we compare different management actions in the same PVA (McCarthy et al. 2003).

Many of the above criticisms were fairly justified to some degree and this realization has resulted in guidelines about how to construct better PVAs and when to apply the technique at all (Ralls et al. 2002, Morris and Doak 2002). Fundamenelly, model choice (in the sense of what complexity should be included) ought to be determined by richness of the available data. A more complex model may be realistic, but the associated gain in accuracy may be more than offset by a decrease in precision (this is the usual bias-variance tradeoff). In addition, in a more complex model we may lack data for many parameters, and this will add uncertainty in (and uneasiness with) the model. In spite of this, we think that PVAs should generally accommodate both process and parameter uncertainty. This can be done formally if data are available or via sensitivity analyses, i.e. by fitting and comparing several models that are based on different assumptions about the magnitude of parameters about which one has no data. If data and knowledge about a population are very scarce, it may not be possible to conduct a reliable PVA, and the benefit of model construction is restricted to a heuristic value. Beissinger and Westphal (1998) very correctly stated that *“doing a credible population viability analysis requires good data and good ecological modeling”*.

10.3. Bayesian population viability analysis

Most PVAs have been carried out using frequentist methods of inference, but some Bayesian PVAs have been developed early on (Wade 2002, Ludwig 1996b) and some have been applied to actual data more recently (Maunder 2004, Green and Bailey 2015, Heard et al. 2013, Hegg et al. 2013, McCaffrey et al. 2012, Hobbs et al. 2015). The benefit of a Bayesian over a frequentist PVA are i) that probability statements about the predicted/projected population is obtained directly, that ii) the propagation of parameter uncertainty into the population projections (and any other estimated quantity) is straightforward and direct and iii) that the inferences can directly be used in formal decision or related analyses. Next we briefly discuss these points.

A fundamental difference between Bayesian and frequentistist inference is the definition of probability. In Bayes probability is used as a general measure of uncertainty about parameters, predictions or any other unknown quantity. Hence, Bayesian PVAs allow one to make statements like *"the probability that population size drops below 3 individuals within 15 years is 0.34"*, *"the probability that the population is in decline is 0.85"* or *"the probability that the population size will be larger in 20 years when management option A is taken compared to when no action is taken is 0.74"*. Such statements are obviously very relevant and are precisely what a manager would like to get from a PVA. By contrast, probability in frequentist inference is defined simply as the long-run frequencies in hypothetical repeated samples of our data set. Any probability statement therefore refers to the data, i.e. how likely is it to observe the such and such data, given some parameters. We can never directly make probability expressions about parameters or other uncertain quantities associated with our model. In a frequentist PVA, we may make statements like *"the probability that the population will drop below 3 individuals within 15 years is 0.34"*. This means that if we could somehow observe and measure, say, 1 million replicate populations and analyse each of them to see how likely they drop below a size of 3, then 340,000 of them would fall below 3 within 15 years. This is not a direct statement about our particular population of interest, as we can do when using Bayesian inference.

As outlined in section 10.2, a reliable PVA should accommodate parameter uncertainty. Propagating parameter uncertainty in a MCMC-based Bayesian PVA is straightforward, as it just needs the simulation of the population dynamics for many different combinations of parameter values drawn from their joint posterior distribution. This results in a posterior distribution of the desired quantity (e.g. population size) which directly includes both parameter and process uncertainty (Ludwig 1996b, Wade 2002). The combination of process stochasticity and uncertainty with parameter uncertainty is straightforward, because they are all expressed in a common currency: with probability distributions. Accounting for parameter uncertainty in classical PVAs is much less straightforward and more difficult, but can be done (Dennis et al. 1991, McGowan et al. 2011, Morris and Doak 2002). Typically, it requires specific simulations that reflect parameter uncertainty and process stochasticity. In some way such simulations are similar in spirit to the Bayesian projection (see e.g. sections XY), but accurately accounting for structure among demographic parameters is very hard. In addition, strictly the obtained quantities have a different meaning, pertaining to some hypothetical replicate data sets and not directly to *your* data set and population. The Bayesian approach may also be more efficient and a comparison of Bayesian and frequentist extinction probabilities showed that the true extinction distribution was better covered in a Bayesian than in a frequentist PVA (Wade 2002).

Finally, a Bayesian PVA yields results that can be used directly for further assessments like decision analyses (Possingham et al. 2002, Drechsler and Burgman 2004, Green and Bailey 2015) or adaptive management (McCarthy and Possingham 2007). These approaches need as input exactly the type of probability statements *about the parameters* (and not about data or functions thereof) that Bayesian PVAs deliver by default. Probability statements about extinction risk are also used for classification in the IUCN risk classification system (Taylor et al. 2002).

10.4. Use of IPM in population viability analysis

The traditional approach to PVA consists of several steps. First, the available demographic or count data are analysed in a piecewise manner (e.g., in different software programs) to estimate demographic parameters and/or population sizes. Second, a stochastic population model is constructed that uses these estimates as input. This model is then used to predict the future

development of the population and to derive relevant quantities such as extinction probabilities. A number of specialised software packages such as VORTEX, RAMAS or ALEX (Lindenmayer et al. 1995) is available to perform these stochastic projections.

In contrast, when you build a Bayesian IPM, you can do all at once or, in different words, when you have built an IPM, then doing a PVA is almost trivially easy. The reason is that an IMP contains already all elements that are essential for PVA: it is composed of a stochastic population model that links demography and population size and that fully accomodates parameter uncertainty. Thus, the IPM already summarizes what we know about the population. The only thing lacking is that in the IPM, the time frame is given by the period over which the data were observed, and to do a PVA we need to make this time frame longer to prediction into the future (or possibly to predict into the past). Thus, the only change that is required when going from an IPM to a formal PVA is that we need to extend the time horizon such that predictions for the future are automatically obtained. This completely obviates the need for any step-wise and piecemeal approach and we can do everything in a single, extremely flexible software, BUGS. This is not only conceptually very elegant and satisfactory, but it has huge practical advantages, especially with respect to error propagation: running a single analysis ensures full and proper propagation into all estimated quantities of all uncertainty coming from process stochasticity, possibly from process uncertainty and from parameter uncertainty. No information is lost, nor do any approximations need to be made, e.g., by summarizing the results from one model as input for another model further down in a stepwise analysis. And, importantly for a user, you don't need to learn a different approach for every step in a complete PVA. If you can construct an IPM in the BUGS language, you will also master its simple use as a tool for a formal PVA – there is only a very small additional step to go as we will soon show.

In BUGS, the application of an IPM for PVA requires that the time loops for the models of the demographic rates and of the population development are extended so that a number of years in the future are covered and the desired quantities can be estimated for those years as well. This results in posterior distributions of demographic rates and population size in these future years and from these, other quantities can be derieved that are typical in PVAs, such as extinction probability or time to extinction. It is also straightforward to compare different management scenarios in terms of their consequences for the demography of the projected population. Each scenario is included in the model by specifying the expected change of demographic rates after the implementation of an action. We then can predict future population sizes under each of these scenarios and compare them directly.

So far, very few published studies have used IPMs for PVA. Maunder (2004) presents several methods used in fisheries stock assessment models than can be used for population viability analysis. He concluded that integrated analyses using Bayesian inference is flexible and powerful to merge different kinds of data (and in general, of knowledge) in a single analysis with proper uncertainty assessment, and that Bayesian IPMs are therefore particularly well suited for PVA. Rhodes *et al.* (2011) studied the impact of different strategies that reduce overall mortality of an endangered population of koalas (*Phascolarctos cinereus*) in Australia. They identified conservation needs by the proportional reduction of each mortality cause that was required to achieve a stable population. Finally, Oppel *et al.* (2014) conducted a classical PVA by estimating the extinction probability of a population of the endangered Montserrat Oriole (*Icterus oberi*).

In the following we show by a simulated data set how a population viability analysis is performed and how we can study the impact of different management actions. By a means of a real data set, we in addition show a more complex example….

10.5. A population viability analysis for the woodchat shrike

We now study the viability of a small, imaginary population of the woodchat shrike that fluctuated between 4 and 20 breeding pairs over 20 study years. The goal of our analysis is to, first, assess the viability of the population by estimating its extinction probability over the next 15 years and, second, to propose efficient conservation interventions that are suitable to reduce the extinction probability. We imagine that during the past 20 years we closely monitored the woodchat shrike population and collected capture-recapture data, recorded the reproductive success of a number of surveyed broods and counted the number of breeding pairs in each year. The data are saved in an R data file and we load these data now:

```
load("Data PVA.Rdata")
```

We introduced the life-history of the woodchat shrike in chapter XY, and the IPM developed here reflects that knowledge. Our model assumes a pre-breeding census and is female-based, i.e. only data on females are included in the model. The process model will include demographic and environmental stochasticity. For the former we use appropriate distributions, while for the latter we include in each demographic rate random effects of the year. For the sake of simplicity, we assume that the temporal variation of the demographic rates is not correlated and that density-dependence is absent in this small population.

We first develop an IPM using the logic explained in chapter 6. Then we need only a few modifications to use the IPM for getting predictions of future population sizes, i.e., to use it in a simple PVA. To do this, we first have to extend the loop of the population model (process model). Using the Markov property of the population model, the stage-specific population sizes are then computed in each year. This depends on the values of the demographic rates for these future years, hence, we also need to extend the loop for the annual demographic rates and a mechanism by which we can propagate information about them forwards in time. We achieve the latter by specifying future demographic rates as being drawn from a Normal distribution that is characterised by a mean and a temporal variance that are both estimated from the data as part of fitting the IPM to the observed data. These minor changes that are required to move from an ordinary IPM to a simple PVA are highlighted in red in the code below. Note that we call *K* the number of years for which we want to make predictions. In the model code we also add a derived quantity `extinct`, which is equal to 1 when the total population size equals zero and 0 otherwise. Thus, this node will produce an estimate of the population extinction rate.

```
# Write BUGS model file
cat(file = "ipm.pvaI.jags", "
model {

# Priors and constraints
mean.logit.sj <- logit(mean.sj)
mean.sj ~ dunif(0, 1)
mean.logit.sa <- logit(mean.sa)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.log.f <- log(mean.f)
mean.f ~ dunif(0, 10)

for (t in 1:(n.occasions-1)){
   p[t] <- mean.p
```

```
   }

for (t in 1:(n.occasions-1+K)){  # here we extend the loop to K more years
   logit.sj[t] <- mean.logit.sj + eps.sj[t]
   eps.sj[t] ~ dnorm(0, tau.sj)
   sj[t] <- ilogit(logit.sj[t])
   logit.sa[t] <- mean.logit.sa + eps.sa[t]
   eps.sa[t] ~ dnorm(0, tau.sa)
   sa[t] <- ilogit(logit.sa[t])
   }

for (t in 1:(n.occasions+K)){   # extended loop also here
   log.f[t] <- mean.log.f + eps.f[t]
   eps.f[t] ~ dnorm(0, tau.f)
   f[t] <- exp(log.f[t])
   }

sigma.sj ~ dunif(0, 10)
tau.sj <- pow(sigma.sj, -2)
sigma.sa ~ dunif(0, 10)
tau.sa <- pow(sigma.sa, -2)
sigma.f ~ dunif(0, 10)
tau.f <- pow(sigma.f, -2)

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# State-space model for count data
# Model for the initial population size
N[1,1] ~ dcat(pNinit[])
N[2,1] ~ dcat(pNinit[])

# Process model over time
for (t in 1:(n.occasions-1+K)){  # note extended loop
   N[1,t+1] ~ dpois(sj[t] * f[t] * (N[1,t] + N[2,t]))
   N[2,t+1] ~ dbin(sa[t], (N[1,t] + N[2,t]))
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t] + N[2,t], tau.obs)
   }

# Poisson regression model for productivity data
for (t in 1:n.occasions){
   J[t] ~ dpois(f[t] * B[t])
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]   # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
   pr.a[t,t] <- sa[t]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j])*prod(q[t:(j-1)])*p[j]
```

```
      pr.a[t,j] <- prod(sa[t:j])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Derived parameters
# Total population size
for (t in 1:(n.occasions+K)){
   Ntot[t] <- N[1,t] + N[2,t]
   }
# Check whether the population goes extinct in future
for (t in 1:K){
   extinct[t] <- equals(Ntot[n.occasions+t], 0)
   }
}
")



# Bundle data
K <- 15                           # Number of years with predictions
bugs.data <- list(marr.j = marr.j, marr.a = marr.a, n.occasions =
ncol(marr.j), rel.j = rowSums(marr.j), rel.a = rowSums(marr.a), J = J, B = B,
count = count, pNinit = disc.unif(1, 50), K = K)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 0.5), mean.sa = runif(1, 0.4,
0.6), mean.f = runif(1, 1.3, 2))}

# Parameters monitored
parameters <- c("mean.sj", "sj", "sigma.sj", "mean.sa", "sa", "sigma.sa",
"mean.p", "mean.f", "f", "sigma.f", "sigma.obs", "N", "Ntot", "extinct")

# MCMC settings
ni <- 3000; nt <- 1; nb <- 1000; nc <- 3

# Call JAGS from R (jagsUI), check convergence and summarize posteriors
ipm.pvaI <- jags(bugs.data, inits, parameters, "ipm.pvaI.jags", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb)
par(mfrow = c(3, 3));  traceplot(ipm.pvaI)
print(ipm.pvaI, 3)

JAGS output for model 'ipm.pvaI.jags', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran for 0.421 minutes at time 2016-08-31 08:24:46.

               mean     sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj       0.284  0.029   0.226   0.283   0.342    FALSE 1 1.003   744
sj[1]         0.231  0.060   0.118   0.231   0.347    FALSE 1 1.001  1946
sj[2]         0.222  0.056   0.116   0.222   0.330    FALSE 1 1.002  1092
[ ...  output truncated... ]
sj[33]        0.293  0.091   0.127   0.287   0.500    FALSE 1 1.001  4134
```

```
sj[34]          0.292  0.091   0.131   0.285   0.505    FALSE 1 1.001  3196
sigma.sj        0.411  0.162   0.141   0.399   0.768    FALSE 1 1.002   888
mean.sa         0.543  0.027   0.490   0.543   0.595    FALSE 1 1.004  1699
sa[1]           0.554  0.050   0.462   0.549   0.670    FALSE 1 1.012  2127
sa[2]           0.538  0.048   0.432   0.538   0.633    FALSE 1 1.008  5028
…
sa[33]          0.544  0.054   0.431   0.543   0.663    FALSE 1 1.011  6000
sa[34]          0.542  0.055   0.427   0.542   0.658    FALSE 1 1.011  6000
sigma.sa        0.160  0.123   0.012   0.132   0.468    FALSE 1 1.048    53
mean.p          0.632  0.033   0.566   0.633   0.694    FALSE 1 1.002   880
mean.f          1.531  0.094   1.348   1.531   1.713    FALSE 1 1.003  2445
f[1]            1.494  0.188   1.093   1.502   1.869    FALSE 1 1.004  1356
f[2]            1.470  0.186   1.070   1.484   1.815    FALSE 1 1.004  1460
…
f[34]           1.547  0.246   1.086   1.535   2.112    FALSE 1 1.006  6000
f[35]           1.546  0.255   1.088   1.530   2.093    FALSE 1 1.018  6000
sigma.f         0.118  0.079   0.003   0.108   0.295    FALSE 1 1.024   441
sigma.obs       2.342  0.907   0.883   2.233   4.426    FALSE 1 1.008   358
N[1,1]          5.702  3.149   1.000   6.000  12.000    FALSE 1 1.005   391
N[2,1]          5.410  3.140   1.000   5.000  12.000    FALSE 1 1.004   466
N[1,2]          4.207  1.783   1.000   4.000   8.000    FALSE 1 1.003   610
N[2,2]          6.544  1.692   3.000   7.000  10.000    FALSE 1 1.002  1417
…
N[1,34]         6.737 16.183   0.000   3.000  38.000     TRUE 1 1.045  3607
N[2,34]         7.567 13.497   0.000   4.000  37.000     TRUE 1 1.012  4987
N[1,35]         6.803 13.618   0.000   3.000  38.000     TRUE 1 1.007  6000
N[2,35]         7.837 16.157   0.000   3.000  42.000     TRUE 1 1.024  4678
Ntot[1]        11.113  2.009   8.000  11.000  15.000    FALSE 1 1.003  4219
Ntot[2]        10.751  1.820   7.000  11.000  14.000    FALSE 1 1.002  2723
…
Ntot[34]       14.304 28.818   0.000   7.000  75.000     TRUE 1 1.026  4037
Ntot[35]       14.640 28.838   0.000   6.000  79.025     TRUE 1 1.015  6000
extinct[1]      0.000  0.000   0.000   0.000   0.000    FALSE 1    NA     1
extinct[2]      0.001  0.036   0.000   0.000   0.000    FALSE 1 1.106  2997
…
extinct[14]     0.207  0.405   0.000   0.000   1.000     TRUE 1 1.002  1613
extinct[15]     0.227  0.419   0.000   0.000   1.000     TRUE 1 1.001  2481
deviance      510.582 15.897 475.451 511.662 538.136    FALSE 1 1.003   637
```

To inspect the results we now produce plots of the predicted population sizes and demographic rates.

```
n.years <- length(ipm.pvaI$mean$Ntot)

par(mfrow = c(2, 2), mar = c(4.5, 4, 1, 1), cex = 1.1)
plot(0, 0, ylim = range(c(ipm.pvaI$q2.5$Ntot, ipm.pvaI$q97.5$Ntot)), xlim =
c(0.5, n.years), ylab = "Total population size", xlab = "Year", las = 1, col =
"black", type = "l", lwd = 2, frame = FALSE, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(1, n.years, 3), labels = seq(1, n.years, 3))
axis(1, at = 1:n.years, labels = rep("", n.years), tcl = -0.25)
polygon(x = c(1:n.years, n.years:1), y = c(ipm.pvaI$q2.5$Ntot,
ipm.pvaI$q97.5$Ntot[n.years:1]), col = "gray90", border = "gray90")
points(ipm.pvaI$q50$Ntot, type = "l", col = "blue", lwd = 2)

plot(ipm.pvaI$mean$f, ylim = range(c(ipm.pvaI$q2.5$f, ipm.pvaI$q97.5$f)), xlim
= c(0.5, n.years), ylab = "Productivity", xlab = "Year", las = 1, col =
"black", type = "l", lwd = 2, frame = FALSE, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(1, n.years, 3), labels = seq(1, n.years, 3))
axis(1, at = 1:n.years, labels = rep("", n.years), tcl = -0.25)
```

```
polygon(x = c(1:n.years, n.years:1), y = c(ipm.pvaI$q2.5$f, ipm.pvaI$q97.5$f
[n.years:1]), col = "gray90", border = "gray90")
points(ipm.pvaI$mean$f, type = "l", col = "blue", lwd = 2)

plot(ipm.pvaI$mean$sj, ylim = range(c(ipm.pvaI$q2.5$sj, ipm.pvaI$q97.5$sj)),
xlim = c(0.5, n.years), ylab = "Juvenile survival", xlab = "Year", las = 1,
col = "black", type = "l", lwd = 2, frame = FALSE, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(1, n.years-1, 3), labels = seq(1, n.years-1, 3))
axis(1, at = 1:(n.years-1), labels = rep("", n.years-1), tcl = -0.25)
polygon(x = c(1:(n.years-1), (n.years-1):1), y = c(ipm.pvaI$q2.5$sj,
ipm.pvaI$q97.5$sj [(n.years-1):1]), col = "gray90", border = "gray90")
points(ipm.pvaI$mean$sj, type = "l", col = "blue", lwd = 2)

plot(ipm.pvaI$mean$sa, ylim = range(c(ipm.pvaI$q2.5$sa, ipm.pvaI$q97.5$sa)),
xlim = c(0.5, n.years), ylab = "Adult survival", xlab = "Year", las = 1, col =
"black", type = "l", lwd = 2, frame = FALSE, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(1, n.years-1, 3), labels = seq(1, n.years-1, 3))
axis(1, at = 1:(n.years-1), labels = rep("", n.years-1), tcl = -0.25)
polygon(x = c(1:(n.years-1), (n.years-1):1), y = c(ipm.pvaI$q2.5$sa,
ipm.pvaI$q97.5$sa [(n.years-1):1]), col = "gray90", border = "gray90")
points(ipm.pvaI$mean$sa, type = "l", col = "blue", lwd = 2)
```

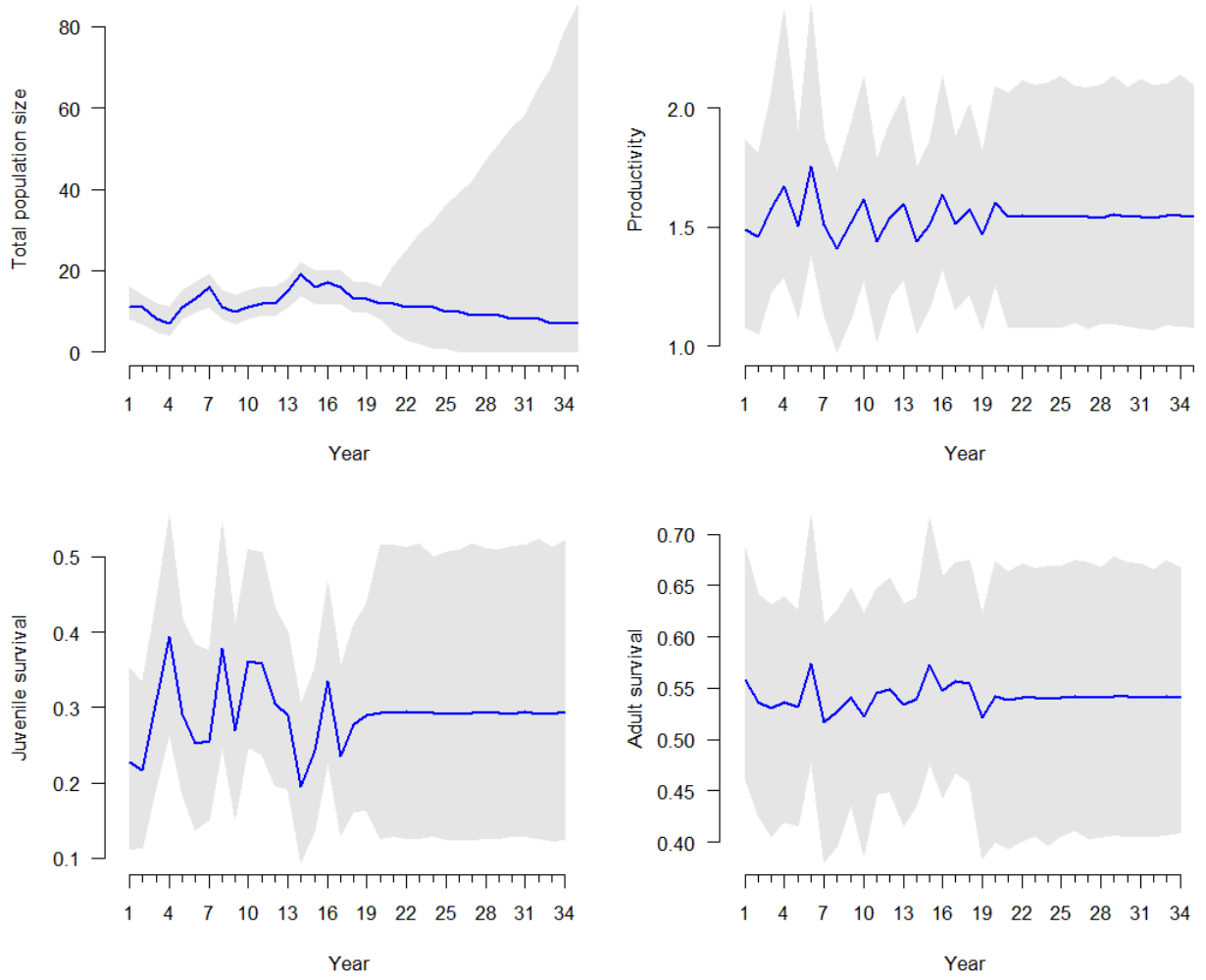


Figure 10.1 Estimated and predicted population size of a woodchat shrike population (top left), and estimated and predicted demographic rates. Blue lines show the posterior median (population size) and posterior means (demographic rates), grey areas indicate 95% credible intervals.

The uncertainty in the predicted population size is increasing very strongly the longer the population is projected into the future (Fig. 10.1). This corresponds to our expectations, since the uncertainty due to stochasticity is accumulating over time in this Markovian process. This is fundamentally different for the demographic rates, where the uncertainty does not change with increasing length of projections. In our model, the demographic rates are independent from one year to the next (there is no Markovian dependence), hence, there is no error propagation over time. The width of the credible intervals remains constant over time and depends on the temporal variability of the corresponding demographic rate and on the estimation uncertainty. If the temporal variability and/or parameter uncertainty was larger, then the credible intervals would also become larger. If we had specified temporal autocorrelation in the demographic rates, then the uncertainty would also increase over time in the prediction period. Note that the uncertainty in the projected quantities does correctly all uncertainty stemming **from** process uncertainty and variability, from demographic uncertainty and from parameter uncertainty. This all occurs "automatically" and we don't have to do anything extra; neat, isn't it?

Next we inspect the extinction probability, which is the traditional inference taken from a PVA (Morris and Doak 2002). As a derived parameter we have calculated the node "extinct" using the BUGS function `equals`. For every future year and MCMC iteration, this function checks whether the total population is equal to zero and returns a 1, otherwise it returns a 0. Hence, taking the mean across all MCMC samples provides an estimate of the extinction probability in a given future year. From the summary output we find that the extinction probability 15 years from now is 0.227. The summary output also provides standard errors and credible intervals of extinction probability, but these are fairly uninformative for a binary indicator variable such as this. As an informative summary, we inspect the cumulative extinction probability over time graphically.

```
plot(ipm.pvaI$mean$extinct, type = "l", ylab = "Cumulative extinction
probability", lwd = 3, xlab = "Year", frame = FALSE, axes = FALSE)
axis(1, at = 1:K, tck = -0.0125, labels = FALSE)
axis(1, at = c(1, 3, 5, 7, 9, 11, 13, 15), labels = c(1, 3, 5, 7, 9, 11, 13,
15), tck = -0.025)
axis(2, las = 1)
abline(h = 0.05, col = "red", lwd = 3)
```

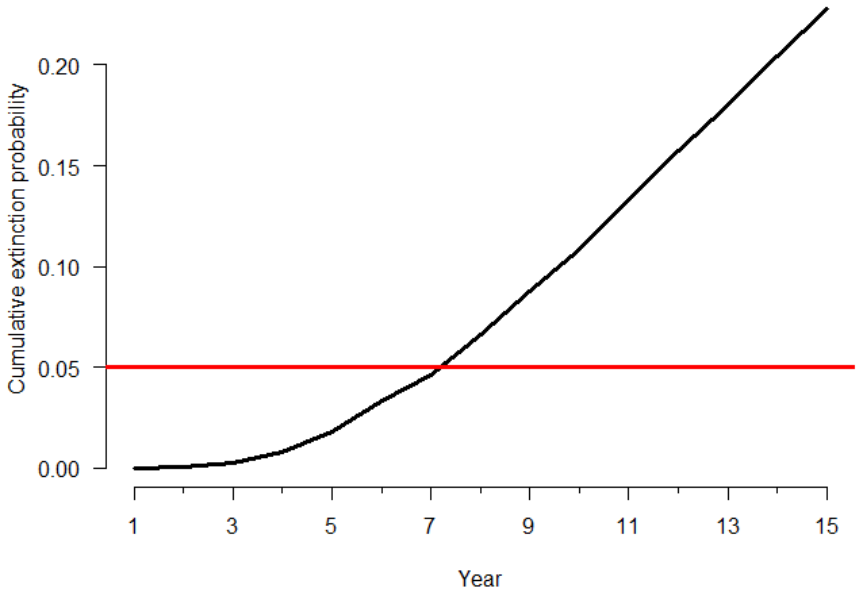


Figure 10.2 Cumulative extinction probability of the woodchat shrike population. The horizontal red line indicates an extinction probability of 0.05. The point where the red and the black lines cross indicates that after 7 years the extinction probability becomes greater than 0.05.

So far we have reported the absolute extinction probability, i.e., we called a population of size zero as extinct. However, it may often be more meaningful to use an extinction threshold (*D*) larger than zero. A population whose size is close to zero is at risk in any case and we may want to have a reasonable faith in that it is secure by choosing an extinction threshold greater than zero. Hence, we can call "quasi extinction" whenever it drops to a size of *D* or smaller. To compute a quasi-extinction probability we could simply change the line in the BUGS code where extinction is defined to this:

```
   extinct[t] <- step(D-Ntot[n.occasions+t])   .
```

Alternatively, we can calculate quasi-extinction (and absolute extinction also) in R based on the JAGS output. Let's calculate the cumulative extinction probabilities for extinction thresholds of D = {1, 3, 5, 7} and produce a plot to compare them.

```
D <- c(1, 3, 5, 7)
T <- length(count)
color <- colorRampPalette(c("blue", "red"))(4)
plot(y = rep(0,K), x = 1:K, type = "n", ylim = c(0, 0.5), ylab = "Quasi-
extinction probability", xlab = "Year", axes = F)
axis(2, las = 1)
axis(1, at = 1:K, tck = -0.0125, labels = FALSE)
axis(1, at = c(1, 3, 5, 7, 9, 11, 13, 15), labels = c(1, 3, 5, 7, 9, 11, 13,
15), tck = -0.025)
qextinct <- matrix(0, ncol = K, nrow = length(ipm.pvaI$sims.list$deviance))
for (i in 1:length(D)){
   qextinct[ipm.pvaI$sims.list$Ntot[,(T+1):(T+K)] <= D[i]] <- 1
   points(apply(qextinct, 2, mean), type = "l", lwd = 2, col = color[i])
```

```
   }
legend("topleft", legend = c("D = 1", "D = 3", "D = 5", "D = 7"), col = color,
lwd = 2, bty = "n")
```

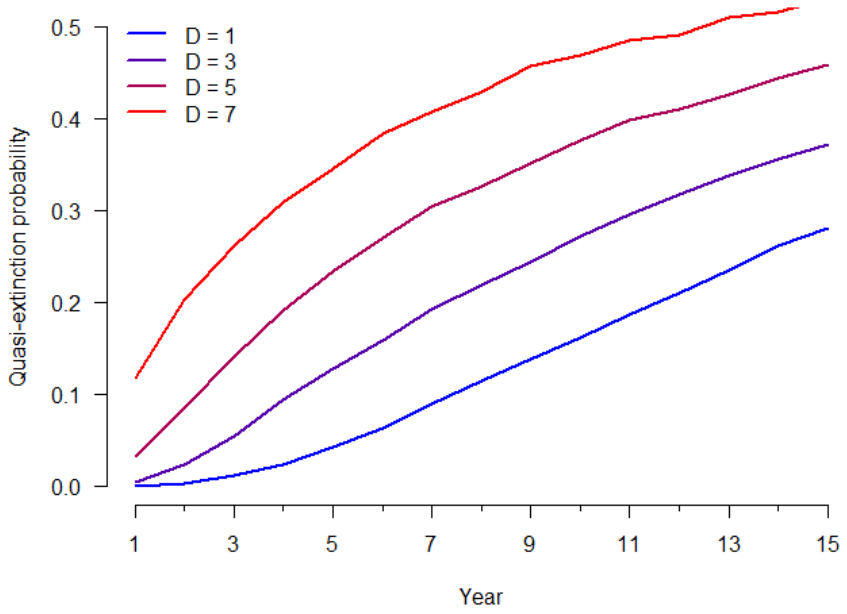


Figure 10.3 Cumulative quasi-extinction probabilities of the woodchat shrike population for different extinction thresholds *D*.

In a Bayesian analysis we represent our knowledge about population size (and other quantities) by posterior distributions, hence, we can use them to make probability statements about these quantities. This is very useful and can provide essential additional insights. For example we can calculate the probability that the population in 15 years will be smaller than the one now simply as follows (note that your result will be slightly different due to the stochastic nature of the MCMC solution):

```
mean(ipm.pvaI$sims.list$Ntot[,T] > ipm.pvaI$sims.list$Ntot[,T+K])
[1] 0.6301
```

Technically, for each MCMC iteration/sample we perform a comparison (i.e. logical test) of the population sizes in years 20 and 35. This logical test returns a value of “FALSE” or a “TRUE” in R, which is interpreted as 0 and 1, respectively, in a calculation. The mean across these binary values therefore returns the probability that the logical test evaluates to true.

A further useful approach is to restrict the calculation of derived quantities on the extinct populations (those that have reached the extinction threshold within the defined time) and to compute the time they needed to go extinct. Time to extinction is not normally distributed (Ludwig 1996a, Lande and Orzack 1988), hence, it is better to express the central tendency of this distribution by the median rather than by the mean. Even more informative is the entire frequency distribution of the time to extinction. Remember that this is the *conditional* distribution of the time to extinction: it is conditional on the extinction event as defined by the

projection time *K* and the chosen extinction threshold *D*. Here we compute the median time to extinction and show a distribution of conditional time to extinction.

```
D <- 3
T <- length(count)
qextinct <- matrix(0, ncol = K, nrow = length(ipm.pvaI$sims.list$deviance))
qextinct[ipm.pvaI$sims.list$Ntot[,(T+1):(T+K)] <= D] <- 1
ext <- which(qextinct[,K]==1)
time.to.extinction <- K+1-apply(qextinct[ext,], 1, sum)
median(time.to.extinction)
[1] 8

par(las = 1)
barplot(table(time.to.extinction), col = "gold", ylab = "Frequency", xlab =
"Conditional time to extinction")
```

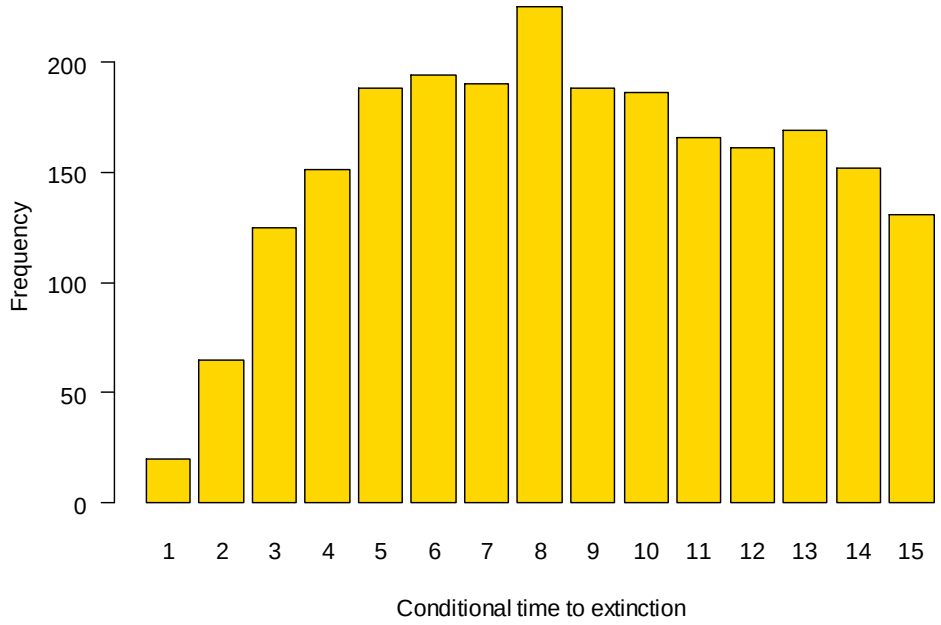


Figure 10.4 Frequency distribution of the conditional time to extinction of the woodchat shrike population. Conditional means that only populations that declined below the extinction threshold of *D* = 3 within *K* = 15 years are included.

Quite often, computing absolute or quasi-extinction risks is not *per se* very interesting or insightful. Rather, a comparison of different management options in terms of the resulting extinction probability, time to extinction or projected population size may be more meaningful. To illustrate, let's assume that we have a set of conservation management options to help the woodchat shrike population and we want to know which one is the most efficient in terms of reducing extinction risk. The options that we could take are:

i) a reduction of nest predation by crows (*Corvidae*) with the effect of a 20% increase in the number of fledglings,

ii) a reduction of the temporal variability of adult survival by half, and

iii) the release of 3 breeding pairs during 5 years, i.e. to perform translocations.

For option 3 we assume that the breeding pairs are released in spring and that they recruit into the population and reproduce immediately. Of course, this means that the population cannot go extinct during the time when releases are conducted. As a control, we compare these options with the "do nothing" option to find out whether any of the three management actions is efficient at all.

Our goal is to write a single model that performs predictions under all four options. Clearly the estimation of the demographic parameters and of the population size from the start of the study until the end is identical for all four scenarios and only the prediction parts of the model differs. Therefore, our code typically uses different loops over time when the past and the future dynamics (3 options) are computed. Thus the model needs a number of adaptations.

```
# Bundle data
K <- 15                              # Number of years with predictions
n.occasions <- ncol(marr.j)          # Number of study years
bugs.data <- list(marr.j = marr.j, marr.a = marr.a, n.occasions = n.occasions,
rel.j = rowSums(marr.j), rel.a = rowSums(marr.a), J = J, B = B, count = count,
pNinit = disc.unif(1, 50), K = K)


# Write BUGS model file
cat(file = "ipm.pvaII.txt", "
model {
# Priors and constraints
mean.logit.sj <- logit(mean.sj)
mean.sj ~ dunif(0, 1)
mean.logit.sa <- logit(mean.sa)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.log.f <- log(mean.f)
mean.f ~ dunif(0, 10)

for (t in 1:(n.occasions-1)){
   p[t] <- mean.p
   }

sigma.sj ~ dunif(0, 10)
tau.sj <- pow(sigma.sj, -2)
sigma.sa ~ dunif(0, 10)
tau.sa <- pow(sigma.sa, -2)
sigma.f ~ dunif(0, 10)
tau.f <- pow(sigma.f, -2)

sigma.obs ~ dunif(0.5, 50)
tau.obs <- pow(sigma.obs, -2)

# Models for demographic rates
# Scenario 1: no change
for (t in 1:(n.occasions-1+K)){
   logit.sj[t] <- mean.logit.sj + eps.sj[t]
   eps.sj[t] ~ dnorm(0, tau.sj)
   sj[t] <- ilogit(logit.sj[t])
   logit.sa[t,1] <- mean.logit.sa + eps.sa[t,1]
   eps.sa[t,1] ~ dnorm(0, tau.sa)
   sa[t,1] <- ilogit(logit.sa[t,1])
   }
for (t in 1:(n.occasions+K)){
   log.f[t,1] <- mean.log.f + eps.f[t,1]
   eps.f[t,1] ~ dnorm(0, tau.f)
   f[t,1] <- exp(log.f[t,1])
   }
```

```
# Scenario 2: increase of productivity
# Past: identical to scenario 1
for (t in 1:n.occasions){
   log.f[t,2] <- log.f[t,1]
   eps.f[t,2] <- eps.f[t,1]
   f[t,2] <- f[t,1]
   }
# Future : increase mean productivity by 20%
for (t in (n.occasions+1):(n.occasions+K)){
   log.f[t,2] <- mean.log.f + log(1.2) + eps.f[t,2]
   eps.f[t,2] ~ dnorm(0, tau.f)
   f[t,2] <- exp(log.f[t,2])
   }

# Scenario 3: reduction of temporal variability in adult survival
# Past: identical to scenario 1
for (t in 1:(n.occasions-1)){
   logit.sa[t,2] <- logit.sa[t,1]
   eps.sa[t,2] <- eps.sa[t,1]
   sa[t,2] <- sa[t,1]
   }
# Future: reduction of temporal variability by half
for (t in n.occasions:(n.occasions-1+K)){
   logit.sa[t,2] <- mean.logit.sa + eps.sa[t,2]
   eps.sa[t,2] ~ dnorm(0, tau.sa*2)     # temporal precision increased
   sa[t,2] <- ilogit(logit.sa[t,2])
   }


# State-space model for count data
N[1,1,1] ~ dcat(pNinit[])
N[2,1,1] ~ dcat(pNinit[])

# Scenario 1: no change
# Process model over time (past and future)
for (t in 1:(n.occasions-1+K)){
   N[1,t+1,1] ~ dpois(f[t,1] * sj[t] * (N[1,t,1] + N[2,t,1]))
   N[2,t+1,1] ~ dbin(sa[t,1], (N[1,t,1] + N[2,t,1]))
   }

# Scenario 2: increase of productivity
# Past
for (t in 1:n.occasions){
   N[1,t,2] <- N[1,t,1]
   N[2,t,2] <- N[2,t,1]
   }
# Future
for (t in n.occasions:(n.occasions-1+K)){
   N[1,t+1,2] ~ dpois(f[t,2] * sj[t] * (N[1,t,2] + N[2,t,2]))
   N[2,t+1,2] ~ dbin(sa[t,1], (N[1,t,2] + N[2,t,2]))
   }

# Scenario 3: decrease of temporal var. in adult survival
# Past
for (t in 1:n.occasions){
   N[1,t,3] <- N[1,t,1]
   N[2,t,3] <- N[2,t,1]
   }
# Future
for (t in n.occasions:(n.occasions-1+K)){
   N[1,t+1,3] ~ dpois(f[t,1] * sj[t] * (N[1,t,3] + N[2,t,3]))
   N[2,t+1,3] ~ dbin(sa[t,2], (N[1,t,3] + N[2,t,3]))
```

```
   }

# Scenario 4: release of 3 females annually during 5 years
# Past
for (t in 1:n.occasions){
   N[1,t,4] <- N[1,t,1]
   N[2,t,4] <- N[2,t,1]
   }
# Future, phase with releases
for (t in n.occasions:(n.occasions+5)){
   N[1,t+1,4] ~ dpois(f[t,1] * sj[t] * (N[1,t,4] + N[2,t,4] + 3))
   N[2,t+1,4] ~ dbin(sa[t,1], (N[1,t,4] + N[2,t,4] + 3))
   }
# Future, after the phase with releases
for (t in (n.occasions+6):(n.occasions-1+K)){
   N[1,t+1,4] ~ dpois(f[t,1] * sj[t] * (N[1,t,4] + N[2,t,4]))
   N[2,t+1,4] ~ dbin(sa[t,1], (N[1,t,4] + N[2,t,4]))
   }

# Observation model
for (t in 1:n.occasions){
   count[t] ~ dnorm(N[1,t,1] + N[2,t,1], tau.obs)
   }

# Poisson regression model for productivity data
for (t in 1:n.occasions){
   J[t] ~ dpois(f[t,1]*B[t])
   }

# Capture-recapture model (multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
   marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
   marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
   }
# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
   q[t] <- 1-p[t]   # Probability of non-recapture
   pr.j[t,t] <- sj[t]*p[t]
   pr.a[t,t] <- sa[t,1]*p[t]
   # Above main diagonal
   for (j in (t+1):(n.occasions-1)){
      pr.j[t,j] <- sj[t]*prod(sa[(t+1):j,1])*prod(q[t:(j-1)])*p[j]
      pr.a[t,j] <- prod(sa[t:j,1])*prod(q[t:(j-1)])*p[j]
      } #j
   # Below main diagonal
   for (j in 1:(t-1)){
      pr.j[t,j] <- 0
      pr.a[t,j] <- 0
      } #j
   } #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
   pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
   pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
   } #t

# Derived parameters
for (t in 1:(n.occasions+K)){
   Ntot[t,1] <- N[1,t,1] + N[2,t,1]     # Total population sizes scenario 1
   Ntot[t,2] <- N[1,t,2] + N[2,t,2]     # Total population sizes scenario 2
   Ntot[t,3] <- N[1,t,3] + N[2,t,3]     # Total population sizes scenario 3
```

```
   Ntot[t,4] <- N[1,t,4] + N[2,t,4]     # Total population sizes scenario 4
   }
}
")


# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 0.5), mean.sa = runif(1, 0.4,
0.6), mean.f = runif(1, 1.3, 2))}

# Parameters monitored
parameters <- c("mean.sj", "sj", "sigma.sj", "mean.sa", "sa", "sigma.sa",
"mean.p", "mean.f", "f", "sigma.f", "N", "Ntot", "sigma.obs")

# MCMC settings
ni <- 3000; nt <- 1; nb <- 1000; nc <- 3

# Call JAGS from R (jagsUI)
ipm.pvaII <- jags(bugs.data, inits, parameters, "ipm.pvaII.txt", n.chains =
nc, n.thin = nt, n.iter = ni, n.burnin = nb)
```

To compare the different scenarios, we compute the quasi-extinction probabilities using a threshold of *D* = 3 for each scenario (note that this represents 3 pairs in our female-based model), and produce a graph with the cumulative quasi-extinction probabilities (Fig. 10.5).

```
D <- 3
T <- length(count)
qextinct1 <- qextinct2 <- qextinct3  <- qextinct4 <- matrix(0, ncol = K, nrow
= length(ipm.pvaII$sims.list$deviance))
qextinct1[ipm.pvaII$sims.list$Ntot[,(T+1):(T+K),1] <= D] <- 1
qextinct2[ipm.pvaII$sims.list$Ntot[,(T+1):(T+K),2] <= D] <- 1
qextinct3[ipm.pvaII$sims.list$Ntot[,(T+1):(T+K),3] <= D] <- 1
qextinct4[ipm.pvaII$sims.list$Ntot[,(T+1):(T+K),4] <= D] <- 1

color <- colorRampPalette(c("blue", "red"))(4)
par(las = 1)
plot(apply(qextinct1, 2, mean), type = "l", ylab = "Quasi-extinction
probability", lwd = 2, xlab = "Year", frame = FALSE, axes = FALSE, col =
color[1])
axis(1, at = 1:K, tck = -0.0125, labels = FALSE)
axis(1, at = c(1, 3, 5, 7, 9, 11, 13, 15), labels = c(1, 3, 5, 7, 9, 11, 13,
15), tck = -0.025)
axis(2)
lines(apply(qextinct2, 2, mean), type = "l", lwd = 2, col = color[2])
lines(apply(qextinct3, 2, mean), type = "l", lwd = 2, col = color[3])
lines(apply(qextinct4, 2, mean), type = "l", lwd = 2, col = color[4])
legend("topleft", lty = rep(1, 4), lwd = rep(2, 4), col = color, legend =
c("Do nothing", "Increased productivity", "Reduced variability",
"Translocation"), bty = "n")
```

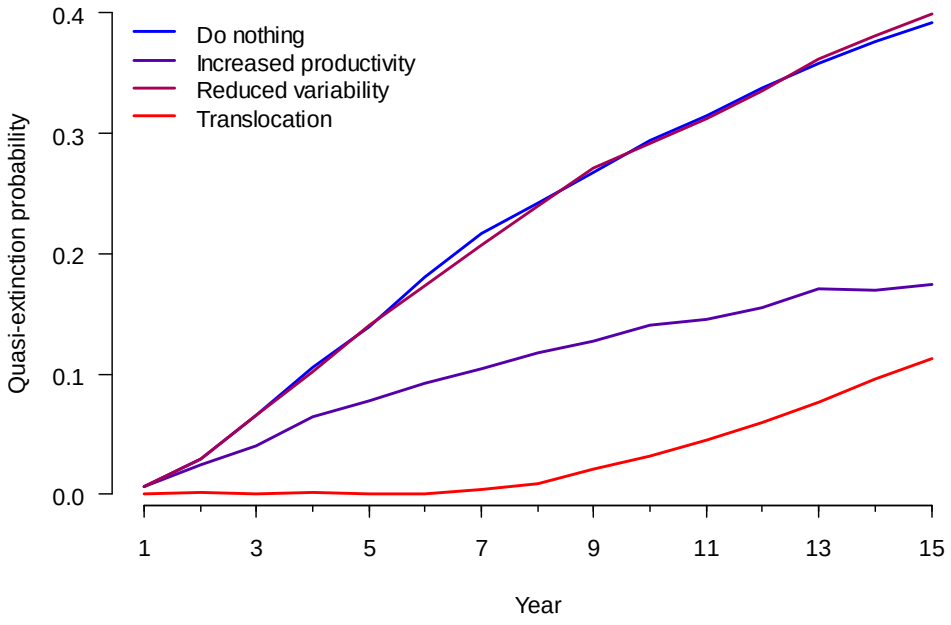


Figure 10.5 Comparison of the cumulative quasi-extinction probabilities of woodchat shrike populations under four different management options (including 'do nothing' as a control).

Translocations appear to be the most promising management option, i.e., the one which reduces most the quasi-extinction probability (Fig. 10.5), followed by the increased productivity. We see that the reduction of the temporal variability of adult survival had almost no effect on the extinction risk and should therefore not be considered. We can rank the different scenarios also in another way by computing the probability that a given scenario with a management action produces a larger population size in 15 years compared to the control scenario where no action is taken.

```
mean(ipm.pvaII$sims.list$Ntot[,n.occasions+K,2] >
ipm.pvaII$sims.list$Ntot[,n.occasions+K,1])
[1] 0.760

mean(ipm.pvaII$sims.list$Ntot[,n.occasions+K,3] >
ipm.pvaII$sims.list$Ntot[,n.occasions+K,1])
[1] 0.449

mean(ipm.pvaII$sims.list$Ntot[,n.occasions+K,4] >
ipm.pvaII$sims.list$Ntot[,n.occasions+K,1])
[1] 0.805
```

When either productivity is increased or individuals are translocated it is quite likely that the population size in 15 years will be larger than when no management action is taken. Since the probabilities are similar for these two option, we can also compare the projected population size under the two options directly and ask the question of what is the probability that translocation results in a larger population size in 15 years compared to predator control.

```
mean(ipm.pvaII$sims.list$Ntot[,n.occasions+K,4] >
ipm.pvaII$sims.list$Ntot[,n.occasions+K,2])
[1] 0.422
```

Thus, although the extinction probability of the population is lower with translocations than with predator control (Fig. 10.5), the projected population size in 15 years hardly differs between these two management options. The reason for that is that translocations only *postpone* extinction, but do not change the extinction risk in the long-term, because the demographic rates remain the same. We can see this by extending the prediction time horizon from 15 to, say, 30 years.

Modeling different scenarios is straightforward, as illustrated above. One difficulty is that we need careful book-keeping for all the different quantities that we calculate. Another difficulty is that we need actual estimates or else reasonable (e.g., "educated") guesses about the magnitude of the change of a demographic rate associated with the implementation of each management action. Sometimes we may not know this and we then want to incorporate into our analysis also the uncertainty in our guess about the magnitude of the effect. Let's assume for example that in the predator control scenario we were uncertain about how much productivity can be increased. Instead of setting a fixed number of 20% increase, we can use a prior for the increase reflecting our uncertainty. We use here a normal distribution centered to the mean value and a precision of 100. The BUGS code is changed in the following way:

```
# Scenario 2: increase of productivity
# Past: identical to scenario 1
for (t in 1:n.occasions){
   log.f[t,2] <- log.f[t,1]
   eps.f[t,2] <- eps.f[t,1]
   f[t,2] <- f[t,1]
   }
# Future : increase mean productivity by 20%
incr.prod ~ dnorm(1.2, 100)
for (t in (n.occasions+1):(n.occasions+K)){
   log.f[t,2] <- mean.log.f + log(incr.prod) + eps.f[t,2]
   eps.f[t,2] ~ dnorm(0, tau.f)
   f[t,2] <- exp(log.f[t,2])
   }
```

Such kind of uncertainty could of course also be included for the other scenarios.

Before we look at an example with an empirical data set, we make a number of comments about the use of an IPM as a tool for conducting a PVA. First, we must not forget that we will not estimate the true population size in most IPM (see chapter XY): when the population counts are subject to false negatives (i.e., detection probability < 1), but we use a Normal, log-Normal or Poisson distribution for the observation model of the counts, the true population size is underestimated and we will clearly overestimate the extinction probability. Even if we estimate population size without bias in an IPM, we may want to use the demographic estimates and apply it to a larger population whose viability shall be assessed. In either case, we need to know (or be able to make an educated guess about) the size of the population whose viability we want to gauge, and to project the future population starting with the given population size. Exercise 10.2 is devoted to this problem.

Second, in an IPM we combine count data with demographic data. Sometimes it may happen that we have only count data or only demographic data.Then, we cannot conduct an IPM as we define it here. However, it may still be possible to conduct a population viability

analysis by adopting either a pure state-space model for counts (see chapter XY) or by using simpler matrix projection methods shown in chapter 3 when data about demographic parameters are available. The major benefit of a Bayesian approach then is that the propagation of the errors due to process variability and parameter uncertainty "happens" automatically.

Tips and troubleshooting

- "Slicer stuck at value with infinite density": This error message can occur when distributions with shape patameters (Beta, Gamma, Dirichlet) are used and a shape parameter is close to zero. The probability mass gets then concentrated to a single point. Adding a small fixed value to the shape parameter can help. We have encountered this message also when a uniform prior for a standard deviation of the form U(0, k) is used. Changing the prior to U(0.001, k) often helped to overcome the problem.
- Model does not converge. Problems may have to do with the observation error for the population counts. Often the following things have to be checked:
    - If the logNormal distribution for the counts is used, a gamma prior for the precision often works well:
      log(popcount) ~ dnorm(N, tau)
      tau ~ dgamma(0.001, 0.001)
    - If the logNormal distribution for the counts is used, and the wish is to use a uniform prior for the standard deviation, care must be taken that the SD gets not too small. Thus the lower limit of the uniform should be larger than zero:
      log(popcount) ~ Normal(N, tau)
      tau <- 1/sigma/sigma
      sigma ~ dunif(0.01, 1)
    - If the Normal distribution for the counts is used, and the wish is to use a uniform prior for the standard deviation, again care must be taken that the SD does not get too small. Again, it must be avoided that the lower limit of the uniform distribution is too low:
      popcount ~ Normal(N, tau)
      tau <- 1/sigma/sigma
      sigma ~ dunif(0.5, 10)

Literature Cited

Abadi, F., O. Gimenez, R. Arlettaz, and M. Schaub. 2010a. An assessment of integrated population models: bias, accuracy, and violation of the assumption of independence. Ecology **91:**7–14.

Abadi, F., O. Gimenez, H. Jakober, W. Stauber, R. Arlettaz, and M. Schaub. 2012. Estimating the strength of density dependence in the presence of observation errors using integrated population models. Ecological Modelling **242:**1–9.

Abadi, F., O. Gimenez, B. Ullrich, R. Arlettaz, and M. Schaub. 2010b. Estimation of immigration rate using integrated population models. Journal of Applied Ecology **47:**393–400.

Allendorf, F. W., and N. Ryman. 2002. The role of genetics in population viability analysis. Pages 50–85 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.

Altwegg, R., A. Roulin, M. Kestenholz, and L. Jenni. 2003. Variation and covariation in survival, dispersal, and population size in barn owls *Tyto alba*. Journal of Animal Ecology **72:**391–399.

Altwegg, R., M. Schaub, and A. Roulin. 2007. Age-specific fitness components and their temporal variation in the barn owl. American Naturalist **169:**47–61.

Aubry, L. M., D. N. Koons, J. Y. Monnat, and E. Cam. 2009. Consequences of recruitment decisions and heterogeneity on age-specfic breeding success in a long-lived seabird. Ecology **90:**2491–2502.

Bayesian population analysis using WinBUGS. A hierarchical perspective.

Beissinger, S. R. 2002. Population viability analysis: past, present, future. Pages 5–17 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.

Beissinger, S. R., and M. I. Westphal. 1998. On the use of demographic models of population viability in endangered species management. Journal of Wildlife Management **62:**821–841.

Béliveau, A., C. J. Schwarz, R. A. Lockhart, M. Schaub, R. Arlettaz, and R. Pradel. 2016. Integrated population modeling: escaping the conventional assumption of independence. Biometrics **999:**999.

Besbeas, P., R. S. Borysiewicz, and B. J. T. Morgan. 2009. Completing the ecological jigsaw. Pages 513–539 *in* D. Thomson, E. Cooch and M. Conroy, editors. Modeling demographic processes in marked populations. Springer, New York.

Besbeas, P., and S. N. Freeman. 2006. Methods for joint inference from panel survey and demographic data. Ecology **87:**1138–1145.

Besbeas, P., S. N. Freeman, and B. J. T. Morgan. 2005. The potential of integrated population modelling. Australian and New Zealand Journal of Statistics **47:**35–48.

Besbeas, P., S. N. Freeman, B. J. T. Morgan, and E. A. Catchpole. 2002. Integrating mark-recapture-recovery and census data to estimate animal abundance and demographic parameters. Biometrics **58:**540–547.

Besbeas, P., J. D. Lebreton, and B. J. T. Morgan. 2003. The efficient integration of abundance and demographic data. Applied Statistics **52:**95–102.

Besbeas, P., and B. J. T. Morgan. 2012. Kalman filter initialisation for integrated population modelling. Applied Statistics **61:**151–162.

Borysiewicz, R. S., B. J. T. Morgan, V. Hénaux, T. Bregnballe, J. D. Lebreton, and O. Gimenez. 2009. An integrated analysis of multisite recruitment, mark-recapture-recovery and multisite

census data. Pages 579–591 *in* D. Thomson, E. Cooch and M. Conroy, editors. Modeling demographic processes in marked populations. Springer, New York.
Boyce, M. S. 1992. Population viability analysis. Annual Review of Ecology and Systematics **23:**481–506.
Brault, S., and H. Caswell. 1993. Pod-specific demography of resident killer whales *Orcinus orca* in British Columbia and Washington State. Ecology **74:**1444–1454.
Brooks, S. P., R. King, and B. J. T. Morgan. 2004. A Bayesian approach to combining animal abundance and demographic data. Animal Biodiversity and Conservation **27.1:**515–529.
Caswell, H. 2000. Prospective and retrospective perturbation analyses: their roles in conservation biology. Ecology **81:**619–627.
Caswell, H. 2001. Matrix population models. Construction, analysis, and interpretation. Sinauer Associates, Sunderland, Massachusetts.
Caswell, H., S. Brault, A. J. Read, and T. D. Smith. 1998. Harbor porpoise and fisheries: an uncertainty analysis of incidental mortality. Ecological Applications **8:**1226–1238.
Caughley, G. 1994. Directions in conservation biology. Journal of Animal Ecology **63:**215–244.
Cohen, J. E. 1979. Ergodic theorems in demography. Bulletin of the American mathematical society **1:**275–295.
Conroy, M. J., and J. P. Carroll. 2009. Quantitative conservation of vertebrates. Wiley-Blackwell, Oxford.
Converse, S. J., W. L. Kendall, P. F. Doherty, and P. G. Ryan. 2009. Multistate models for estimation of survival and reproduction in the grey-headed albatross (*Thalassarche chrysostoma*). The Auk **126:**77–88.
Courchamp, F., T. Clutton-Brock, and B. T. Grenfell. 1999. Inverse density dependence and the Allee effect. Trends in Ecology and Evolution **14:**405–410.
Dennis, B., P. L. Munholland, and J. M. Scott. 1991. Estimation of growth and extinction parameters for endangered species. Ecological Monographs **61:**115–143.
Dennis, B., J. M. Ponciano, S. R. Lele, M. L. Taper, and D. F. Staples. 2006. Estimating density dependence, process noise, and observation error. Ecological Monographs **76:**323–341.
Dennis, B., and M. L. Taper. 1994. Density dependence in time series observations of natural populations: estimation and testing. Ecological Monographs **64:**205–224.
Drechsler, M., and M. A. Burgman. 2004. Combining population viability analysis with decision analysis. Biodiversity and Conservation **13:**115–139.
Ezard, T. H. G., J. M. Bullock, H. J. Dalgleish, A. Millon, F. Pelletier, A. Ozgul, and D. N. Koons. 2010. Matrix models for a changeable world: the importance of transient dynamics in population management. Journal of Applied Ecology **47:**515–523.
Ferrière, R., F. Sarrazin, S. Legendre, and J. P. Baron. 1996. Matrix population models applied to viability analysis and conservation: theory and practice using the ULM software. Acta Oecologica **17:**629–656.
Fieberg, J., and S. P. Ellner. 2000. When is it meaningful to estimate an extinction probability? Ecology **81:**2040–2047.
Fieberg, J., and S. P. Ellner. 2001. Stochastic matrix models for conservation and management: a comparative review of methods. Ecology Letters **4:**244–266.
Forslund, P., and T. Pärt. 1995. Age and reproduction in birds - hypotheses and tests. Trends in Ecology and Evolution **10:**374–378.
Gaillard, J. M., D. Allaine, D. Pontier, N. G. Yoccoz, and D. E. L. Promislow. 1994. Senescence in natural populations of mammals: a reanalysis. Evolution **48:**509–516.

Gaillard, J. M., M. Festa-Bianchet, N. G. Yoccoz, A. Loison, and C. Toigo. 2000. Temporal variation in fitness components and population dynamics of large herbivores. Annual Review of Ecology and Systematics **31:**367–393.

Gaillard, J. M., N. G. Yoccoz, J. D. Lebreton, C. Bonenfant, S. Devillard, A. Loison, D. Pontier, and D. Allaine. 2005. Generation time: a reliable metric to measure life-history variation among mammalian populations. American Naturalist **166:**119–123.

Gauthier, G., P. Besbeas, J. D. Lebreton, and B. J. T. Morgan. 2007. Population growth in snow geese: a modeling approach integrating demographic and survey information. Ecology **88:**1420–1429.

Gauthier, G., E. Milot, and H. Weimerskirch. 2012. Estimating dispersal, recruitment and survival in a biennially breeding species, the Wandering Albatross. Journal of Ornithology **152:**457–467.

Gill, J. A., K. Norris, P. M. Potts, T. G. Gunnarsson, P. W. Atkinson, and W. J. Sutherland. 2001. The buffer effect and large-scale population regulation in migratory birds. Nature **412:**436–438.

Gilpin, M. E., and M. E. Soulé. 1986. Minimum viable populations: processes of species extinction. Pages 19–34 *in* M. Soulé, editor. Conservation biology: the science of scarcity and diversity. Sinauer, Sunderland.

Ginzburg, L. R., S. Ferson, and H. R. Akcakay. 1990. Reconstructibility of density dependence and the conservative assessment of extinction risks. Conservation Biology **4:**63–70.

Gould, W. R., and J. D. Nichols. 1998. Estimation of temporal variability of survival in animal populations. Ecology **79:**2531–2538.

Green, A. W., and L. L. Bailey. 2015. Using Bayesian population viability analysis to define relevant conservation objectives. Plos ONE **10:**e0144786.

Hadley, G. L., J. J. Rotella, R. A. Garrott, and J. D. Nichols. 2006. Variation in probability of first reproduction of Weddell seals. Journal of Animal Ecology **75:**1058–1070.

Harvey, P. H., and R. M. Zammuto. 1985. Patterns of mortality and age at first reproduction in natural populations of mammals. Nature **315:**319–320.

Hastings, A. 1997. Population biology. Springer, New York.

Heard, G. W., M. A. McCarthy, M. P. Scroggie, J. B. Baumgartner, and K. M. Parris. 2013. A Bayesian model of metapopulation viability, with application to an endangered amphibian. Diversity and Distributions **19:**555–566.

Hegg, D., D. I. MacKenzie, and I. G. Jamieson. 2013. Use of Bayesian population viability analysis to assess multiple management decisions in the recovery programme for the Endangered takahe *Porphyrio hochstetteri*. Oryx **47:**144–152.

Hobbs, N. T., C. Geremia, J. Treanor, R. Wallen, P. J. White, M. B. Hooten, and J. C. Rhyan. 2015. State-space modeling to support management of brucellosis in the Yellowstone bison population. Ecological Monographs **85:**525–556.

Hostetler, J. A., T. S. Sillett, and P. P. Marra. 2015. Full-annual-cycle population models for migratory birds. The Auk **132:**433–449.

Jones, J. H. 2007. demogR: A package for the construction and analysis of age-structured demographic models in R. Journal for Statistical Software **22:**1–28.

Keller, L. F. 1998. Inbreeding and its fitness effects in an insular population of song sparrows. Evolution **52:**240–250.

Kellner, K. F. 2014. Run JAGS (specifically, libjags) from R; an alternative user interface for rjags.

Kramer, A. M., B. Dennis, A. M. Liebhold, and J. M. Drake. 2009. The evidence for Allee effects. Population Ecology **51:**341–354.

Lande, R. 1988. Genetics and demography in biological conservation. Science **241:**1455–1460.

Lande, R. 1993. Risks of population extinction from demographic and environmental stochasticity and random catastrophes. American Naturalist **142:**911–927.
Lande, R., S. Engen, and B. E. Saether. 2003. Stochastic population dynamics in ecology and conservation. Oxford University Press, Oxford.
Lande, R., and S. H. Orzack. 1988. Extinction dynamics of age-structured populations in a fluctuating environmenat. Proceedings of the National Academy of Sciences of the United States of America **85:**7418–7421.
Lebreton, J. D. 2005. Dynamical and statistical models for exploited populations. Australian and New Zealand Journal of Statistics **47:**49–63.
Lefkovitch, L. P. 1965. The study of population growth in organisms grouped by stages. Biometrics **21:**1–18.
Lefranc, N., and T. Worfalk. 1997. Shrikes. A guide to the shrikes of the world. Pica Press, East Sussex.
Legendre, S., and J. Clobert. 1995. ULM, a software for conservation and evolutionary biologists. Journal of Applied Statistics **22:**817–834.
Leslie, P. H. 1945. On the use of matrices in certain population mathematics. Biometrika **33:**183–212.
Lindenmayer, D. B., M. A. Burgman, H. R. Akcakay, R. C. Lacy, and H. P. Possingham. 1995. A review of the generic computer programs ALEX, RAMAS/space and VORTEX for modelling the viability of wildlife metapopulations. Ecological Modelling **82:**161–174.
Link, W. A. 1999. Modeling pattern in collections of parameters. Journal of Wildlife Management **63:**1017–1027.
Link, W. A., and R. J. Barker. 2005. Modeling association among demographic parameters in analysis of open population capture-recapture data. Biometrics **61:**46–54.
Loison, A., M. Festa-Bianchet, J. M. Gaillard, J. T. Jorgenson, and J.-M. Jullien. 1999. Age-specific survival in five populations of ungulates: evidence of senescence. Ecology **80:**2539–2554.
Ludwig, D. 1996a. The distribution of population survival times. American Naturalist **147:**506–525.
Ludwig, D. 1996b. Uncertainty and assessment of extinction probabilities. Ecological Applications **6:**1067–1076.
Ludwig, D. 1999. Is it meaningful to estimate a probability of extinction? Ecology **80:**298–310.
Mace, G. M., N. J. Collar, K. J. Gaston, C. Hilton-Taylor, H. R. Akcakay, N. Leader-Williams, E. J. Milner-Gulland, and S. N. Stuart. 2008. Quantification of extinction risk: IUCN's system for classifying threatened species. Conservation Biology **22:**1424–1442.
Maunder, M. N. 2004. Population viability analysis based on combining Bayesian, integrated, and hierarchical analyses. Acta Oecologica **26:**85–94.
McCaffrey, R., A. Solonen, and E. Crone. 2012. Frog population viability under present and future climate conditions: a Bayesian state-space approach. Journal of Animal Ecology **81:**978–985.
McCarthy, M. A., S. J. Andelman, and H. P. Possingham. 2003. Reliability of relative predictions in population viability analysis. Conservation Biology **17:**982–989.
McCarthy, M. A., and H. P. Possingham. 2007. Active adaptive management for conservation. Conservation Biology **21:**956–963.
McCrea, R. S., B. J. T. Morgan, O. Gimenez, P. Besbeas, T. Bregnballe, V. Hénaux, and J. D. Lebreton. 2010. Multi-site integrated population modelling. Journal of Agricultural, Biological, and Environmental Statistics **15:**539–561.
McGowan, C. P., M. C. Runge, and M. A. Larson. 2011. Incorporating parameteric uncertainty into population viability analysis models. Biological Conservation **144:**1400–1408.

Middleton, D. A. J., and R. M. Nisbet. 1997. Population persistence time: estimates, models, and mechanisms. Ecological Applications **7:**107–117.
Mills, L. S. 2013. Conservation of wildlife populations. Demography, Genetics and Management. Wiley-Blackwell, West Sussex.
Mills, L. S., S. G. Hayes, C. Baldwin, M. J. Wisdom, J. Citta, D. J. Mattson, and K. Murphy. 1996. Factors leading to different viability predictions for a grizzly bear data set. Conservation Biology **10:**863–873.
Morris, W. F., and D. F. Doak. 2002. Quantitative conservation biology. Sinauer, Sunderland, Massachusetts, USA.
Newton, I. 1998. Population limitation in birds. Academic Press, London.
Newton, I. 2004. Population limitation in migrants. Ibis **146:**197–226.
Oli, M. K., and F. S. Dobson. 2003. The relative importance of life-history variables to population growth rate in mammals: Cole's prediction revisited. American Naturalist **161:**422–440.
Oppel, S., G. Hilton, N. Ratcliffe, C. Fenton, J. Daley, G. Gray, J. Vickery, and D. Gibbons. 2014. Assessing population viability while accounting for demographic and environmental uncertainty. Ecology **95:**1809–1818.
Pasinelli, G., M. Schaub, G. Hafliger, M. Frey, H. Jakober, M. Muller, W. Stauber, P. Tryjanowski, J.-L. Zollinger, and L. Jenni. 2011. Impact of density and environmental factors on population fluctuations in a migratory passerine. Journal of Animal Ecology **80:**225–234.
Péron, G., P.-A. Crochet, P. F. Doherty, and J. D. Lebreton. 2010. Studying dispersal at the landscape scale: efficient combination of population surveys and capture-recapture data. Ecology **91:**3365–3375.
Pilastro, A., G. Tavecchia, and G. Marin. 2003. Long living and reproduction skipping in the fat dormouse. Ecology **84:**1784–1792.
Possingham, H. P., D. B. Lindenmayer, and G. N. Tuck. 2002. Decision theory for population viability analysis. Pages 470–489 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.
Powell, L. A. 2007. Approximating variance of demographic parameters using the delta method: a reference for avian biologists. The Condor **109:**949–954.
Pradel, R., A. R. Johnson, A. Viallefont, R. G. Nager, and F. Cézilly. 1997. Local recruitment in the Greater Flamingo: a new approach using capture-recapture data. Ecology **78:**1431–1445.
Pradel, R., and J. D. Lebreton. 1999. Comparison of different approaches to the study of local recruitment of breeders. Bird Study **46:**74–81.
Ralls, K., S. R. Beissinger, and J. F. Cochrane. 2002. Guidelines for using population viability analysisin endangered-species management. Pages 521–550 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.
Reed, J. M., L. S. Mills, J. B. Dunning, E. S. Menges, K. S. McKelvey, R. Frye, S. R. Beissinger, M. C. Anstett, and P. Miller. 2002. Emerging issues in population viability analysis. Conservation Biology **16:**7–19.
Rhodes, J. R., C. F. Ng, D. L. de Villiers, H. J. Preece, C. A. McAlpine, and H. P. Possingham. 2011. Using integrated population modelling to quantify the implications of multiple threatening processes for a rapidly declining population. Biological Conservation **144:**1081–1088.
Ridout, M. S., and P. Besbeas. 2004. An empirical model for underdispersed count data. Statistical Modelling **4:**77–89.
Rodenhouse, N. L., T. W. Sherry, and R. T. Holmes. 1997. Site-dependent regulation of population size: a new synthesis. Ecology **78:**2025–2042.

Rodenhouse, N. L., T. S. Sillett, P. J. Doran, and R. T. Holmes. 2003. Multiple density-dependence mechanisms regulate a migratory bird population during the breeding season. Proceedings in the Royal Society of London Series.B Biological Sciences **270:**2105–2110.

Sanderson, F. J., P. F. Donald, D. J. Pain, I. J. Burfield, and F. P. J. van Bommel. 2006. Long-term population declines in Afro-Palaearctic migrant birds. Biological Conservation **131:**93–105.

Schaub, M., and D. Fletcher. 2015. Estimating immigration using a Bayesian integrated population model: choice of parametrization and priors. Environmental and Ecological Statistics **22:**535–549.

Schaub, M., J. von Hirschheydt, and M. U. Gruebler. 2015. Differential contribution of demographic rate synchrony to population synchrony in barn swallows. Journal of Animal Ecology **84:**1530–1541.

Schaub, M., H. Jakober, and W. Stauber. 2013. Strong contribution of immigration to local population regulation: evidence from a migratory passerine. Ecology **94:**1828–1838.

Shaffer, M. L. 1981. Minimum population sizes for species conservation. BioScience **31:**131–134.

Sillett, T. S., N. L. Rodenhouse, and R. T. Holmes. 2004. Experimentally reducing neighbor density affects reproduction and behavior of a migratory songbird. Ecology **85:**2467–2477.

Skalski, J. R., K. E. Ryding, and J. J. Millspaugh. 2005. Wildlife demography. Elsevier.

Stubben, C., and B. Milligan. 2007. Estimating and analyzing demographic models using the popbio package in R. Journal of Statistical Software **22:**1–23.

Szostek, K. L., M. Schaub, and P. H. Becker. 2014. Immigrants are attracted by local pre-breeders and recruits in a seabird colony. Journal of Animal Ecology **83:**1015–1024.

Tavecchia, G., P. Besbeas, T. Coulson, B. J. T. Morgan, and T. H. Clutton-Brock. 2009. Estimating population size and hidden demographic parameters with state-space modeling. The American naturalist **173:**722–733.

Tavecchia, G., R. Pradel, V. Boy, A. R. Johnson, and F. Cézilly. 2001. Sex- and age-related variation in survival and cost of first reproduction in greater Flamingos. Ecology **82:**165–174.

Taylor, B. L., P. R. Wade, U. Ramakrishnan, M. Gilpin, and H. R. Akcakay. 2002. Incorporating uncertaiity in population viability analyses for the purpose of classifying species by risk. Pages 239–252 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.

Tuljapurkar, S. D., and S. H. Orzack. 1980. Population dynamics in variable environments. 1. Long-run growth rates and extinction. Theoretical Population Biology **18:**314–342.

Valpine, P. de. 2012. Frequentist analysis of hierarchical models for population dynamics and demographic data. Journal of Ornithology **152:**393–408.

Wade, P. R. 2002. Bayesian population viability analysis. Pages 213–238 *in* S. Beissinger and D. McCullough, editors. Population viability analysis. The University of Chicago Press, Chicago.