Demographic methods in life history theory

An introduction with R

Yngvild Vindenes (yngvild.vindenes@ibv.uio.no)

Department of Biosciences, University of Oslo.

Contents

Preface **5**

1 Exponential growth **7**
1.1 Learning goals . . . 7
1.2 The exponential growth model . . . 7
1.3 Exponential growth in structured populations . . . 8
1.4 The role of exponential growth in ecology and evolution . . . 9
1.5 Exercises . . . 9

2 Life tables **11**
2.1 Learning goals . . . 11
2.2 Age-specific survival and fecundity . . . 12
2.3 Life history strategies and trade-offs . . . 13
2.4 A basic life table . . . 14
2.5 Dynamic and static life tables . . . 16
2.6 Life table calculations . . . 17
2.7 Fitness measures in life history theory . . . 22
2.8 Semelparity and iteroparity . . . 23
2.9 The Lack clutch . . . 24
2.10 Exercises . . . 25

3 Age-structured matrix population models **29**
3.1 Learning goals . . . 29
3.2 Age-structured growth . . . 30
3.3 Pre- and post-reproductive census . . . 32
3.4 Life cycle graph . . . 36
3.5 Projected growth . . . 36
3.6 Asymptotic properties . . . 39
3.7 Sensitivity and elasticity of λ . . . 46
3.8 Senescence . . . 53
3.9 Life table response experiments (LTRE) . . . 56
3.10 Exercises . . . 60

4 Stage structured models **63**

4.1 Learning goals . . . 63
4.2 General stage structured matrix model . . . 63
4.3 Decomposing **U** and **F** . . . 65
4.4 Examples of stage structured models . . . 66
4.5 Integral projection models (IPMs) . . . 73
4.6 Exercises . . . 74

5 Stochasticity **79**
5.1 Learning goals . . . 79
5.2 Variable vs unpredictable environments . . . 79
5.3 The Lewontin-Cohen model . . . 80
5.4 Stochastic growth of structured populations . . . 90
5.5 Exercises . . . 102

Bibliography **105**

A Suggested solutions to exercises **107**
A.1 Chapter 1 exercises . . . 107
A.2 Chapter 2 exercises . . . 110
A.3 Chapter 3 exercises . . . 115
A.4 Chapter 4 exercises . . . 127
A.5 Chapter 5 exercises . . . 138

B R code collection **149**
B.1 Life tables . . . 149
B.2 Matrix population models . . . 150
B.3 Stochastic models . . . 154
B.4 Parameters and matrices for the songbird example . . . 158

Preface

This compendium provides an introduction to demographic methods in life history theory, including R code for main calculations, and is developed for the course BIOS5112/9112 at the University of Oslo.

Learning goals are indicated in the beginning of each chapter. Throughout the text, particularly important concepts are highlighted in **bold type**. The compendium is written mainly to be read online, but you can also download a pdf or ebook version (use download button on top).

Before the widespread availability of computers, demographic methods consisted largely of manual calculations using life tables. However, the current computational tools and software like R [R Core Team, 2022] make life table calculations more available than before, and have also made matrix population models widely accessible. Matrix population models generalize the life table approach to other kinds of structure than age structure, and have a wide range of applications.

This compendium starts with a brief chapter on exponential growth, which is a foundation to understand the methods and models of later chapters. Chapter 2 introduces life tables for age structured populations, before matrix models are introduced in the following chapters. The last chapter (chapter 5) provides a short introduction to stochastic population models and evolution of bet-hedging strategies.

For those who are interested in learning matrix models more thoroughly, Caswell [2001] provides a detailed introduction for biologists.

The compendium is still work in progress - if you find errors or have questions / suggestions, please send an email (yngvild.vindenes@ibv.uio.no).

Using R in the compendium

Readers are assumed to have a basic knowledge of R, including how to construct vectors, matrices and data frames, and basic knowledge of how R functions work. The compendium includes examples throughout the text, and exercises after each chapter with suggested solutions in appendix A. Note that some

functions from different chapters build on previously defined functions, so you cannot apply them unless previously defined functions are also uploaded in R. All R functions are collected in appendix B, and here you can also download an RData-file with all functions from the compendium (and relevant R objects for one of the repeated examples).

The syntax used is a mix of base R syntax and 'tidyverse' syntax for manipulating data frames with 'dplyr', and we use ggplot2 for plotting [Wickham et al., 2019]. To run the code in this compendium you should install and load a few packages:

```
library(tidyverse)#Tidyverse packages
library(MASS)#Some statistical functions
library(MetBrewer)#Color palettes for plotting

#Define color vectors for plotting:
colors2 <- met.brewer(2, name = "NewKingdom", type = "discrete")
colors3 <- met.brewer(3, name = "Derain", type = "discrete")
colors4 <- met.brewer(4, name = "Hokusai1", type = "discrete")
colors5 <- met.brewer(5, name = "Hokusai2", type = "discrete")
```

License

Chapter 1

Exponential growth

Many of the classical population models in ecology (e.g. logistic growth, Lotka-Volterra models for predation and competition) build on the implicit assumption that all individuals are equal: At any given time they all have the same chances of survival and reproduction. Such models are a good starting point for understanding population dynamics, and provide a useful baseline for coparison to more complex models. The simplest model for population growth is the **exponential growth model** (also known as the geometric growth model), applying to a population in a constant environment, with no density dependence.

This chapter will cover some basic properties of the exponential growth model. This is a baseline model in much of ecological and evolutionary theory, and provides an important starting point for understanding growth and fitness in structured populations as well.

1.1 Learning goals

- Describe the exponential model for population growth.
- Discuss the importance of this model in ecology and evolutionary biology.
- Do simple calculations and projections in R.

1.2 The exponential growth model

If N_t is the population size at time t, then the next year's population size is given by

$$N_{t+1} = \lambda N_t. \tag{1.1}$$

This model assumes that the population grows with a *constant rate* λ each year. If $\lambda > 1$, the population will increase over time, if $\lambda = 1$ it remains constant (static), and if $\lambda < 1$ the population will decline over time. After t years, a population starting at size N_0 is given by

$$N_t = \lambda^t N_0 = e^{rt} N_0. \tag{1.2}$$

The parameter $r = \ln \lambda$ is the instantaneous growth rate, sometimes called the *Malthusian growth rate* after the early work of Thomas Malthus [1798] on human population growth.

On the logarithmic scale (log scale), we get

$$\ln N_t = t \ln \lambda + \ln N_0 = rt + \ln N_0. \tag{1.3}$$

Since the exponential growth model is linear on log scale, log scale is often used to study population growth and how the growth rate r depends on environmental variables.

1.3 Exponential growth in structured populations

The exponential growth model presented above implies that all individuals have the same survival and reproductive output each time step. However, individuals typically differ in a range of properties, such as age, size, and sex, potentially affecting their survival probability and reproductive success. Demography provides the link between individual variation in survival and reproduction arising from such differences, and population level processes like population growth. The matrix population model for a constant environment (covered in chapters 3 and 4) extends the exponential growth model to structured populations. As long as the parameters governing survival and reproduction in the different stages (i.e. population groups representing differences due to e.g. age) are constant over time, it turns out that structured populations do, after some initial fluctuations, grow exponentially at a constant growth rate λ (see chapter 3).

1.4 The role of exponential growth in ecology and evolution

The exponential growth model is simple, yet plays an important role in ecological and evolutionary theory. It assumes no density regulation, which cannot be true for any population over longer time intervals. However, the purpose of the exponential growth model is generally *not* to describe exact population growth over time, but rather to describe the *growth potential* of a given population, or a subpopulation with a given phenotype, genotype or allele. In that case, the growth rate λ is a measure of the fitness of the subpopulation. For small populations, like the initial population of an invasive species, exponential growth often provides a good description of population growth during the critical initial phase when the success of the invasion is usually determined. Similarly, the exponential growth model is useful to predict the ultimate fate of a new mutation (the chances that it becomes fixed). Mutations tend to arise in one individual at a time, and the subpopulation of mutants will therefore be small at first. Due to chance events even beneficial mutations are often lost in this critical initial phase. If the growth rate of this mutant population is positive and higher than that of the wild-type sub-population, the allele has a chance to spread and become fixed in the population. To describe the exact dynamics of allele frequencies in a population over time requires more complex models that account for density and frequency dependence.

1.5 Exercises

For suggested solutions, see appendix A.

Exercise 1.1

1. Write an R function that takes λ, N_0 (initial population size) and number of time steps T as inputs, and returns a vector of population sizes $[N_0, N_1, ..., N_T]$ based on exponential growth.
2. Use the function to make some plots showing population growth for different values of $\lambda < 0$, $\lambda = 1$, and $\lambda > 1$, for the same initial population size.

Exercise 1.2

To your dismay, you discover that a population of long-tailed silverfish ("Skjeggkre", *Ctenolepisma longicaudata*) has moved into your apartment. After some

Figure 1.1: Long-tailed silverfish (*Ctenolepisma longicaudata*). Photo: Johan Mattson.

initial investigations, you find out that the population grows by approximately 1% each day.

1. What is the value of λ for this population? What is the time unit?
2. How long will it take for the population to double in size?
3. If the current population size is 100 individuals, how large will it be after one year - assuming nothing is stopping its growth? Make a plot of the population growth over this period.

Chapter 2

Life tables

A **life table** is an important tool in demography, helping us keep track of birth and death rates of individuals across age or age groups. Life tables are used extensively in human demography, social sciences, and epidemiology, for instance to calculate the remaining life expectancy at each age, and to project future mortality rates. In life history theory, which aims to understand the evolution of different life history strategies, life tables are used to calculate important parameters for ecological and evolutionary dynamics, such as the net reproductive rate R_0 and the long-term population growth rate λ.

Life tables are closely connected to age-structured matrix models and life cycle graphs (chapter 3) - these are all different representations of the life history of a given population. While life tables are useful for manual calculations, matrix models are the most efficient way to do life history calculations, and have several additional useful properties. Matric models are also able to describe with many different kinds of structure beyond age structure.

2.1 Learning goals

- Define life tables, explain the difference between a dynamic and static life table.
- Perform simple life table calculations (e.g. net reproductive rate R_0, generation time T_C).
- Write the Euler-Lotka equation for a given life table and explain how it can be solved to find the long-term growth rate λ.
- Discuss the advantages and disadvantages of different fitness measures.

- Explain how life tables can be used to study the evolution of different life history strategies, including semelparity/iteroparity and clutch size.

2.2 Age-specific survival and fecundity

Before going into the life table methods in detail, this section will introduce the basic concepts of age-specific survival and fecundity, as well as a brief discussion of life history strategies.

One of the most basic ways to describe the life history of a population, is through the age-specific probability of survival and fecundity (average number of offspring produced per individual). The **survivorship** l_x is the survival probability from birth (age 0) to age x, where $l_0 = 1$ by definition. If we follow a cohort of individuals from their birth until all are dead, and record the number of individuals n_x still alive in the cohort at each age x, then l_x is given by n_x/n_0.

If we plot l_x against age x, this curve will always (eventually) decline to zero, as no individual can survive forever (although some species are almost immortal). However, the way in which the curve declines contains important information about the life history of the population. It is common to distinguish between three main types of survivorship curves (on log scale, fig 2.1): With *type I* curves, $\ln l_x$ declines very slowly for young ages, but rapidly at old age. This kind of survival is typical for many long-lived mammals, including humans and elephants. With *type II* curves, $\ln l_x$ declines linearly with age, which means that the mortality is constant and age-independent. This pattern is typical for some birds and smaller mammals. With *type III* , the $\ln l_x$ curve declines very rapidly in the beginning (for the youngest ages), and then slowly, which is a typical pattern for many trees (producing seeds that have high mortality rates) and fish.

The **annual survival probability** p_x is the probability of survival from from age x to age $x + 1$. The survivorship probability l_x is related to the annual survival probability p_x as

$$\begin{aligned} l_0 &= 1 \\ l_x &= l_{x-1} p_{x-1}, \ x \geq 1. \end{aligned} \tag{2.1}$$

The **fecundity** m_x is the number of newborn offspring produced by an individual at age x, and fecundity curves (plotting m_x against age x) can also contain information about the general life history of the species. For instance, in many mammals fecundity reaches a peak and then declines with age, while in many size-structured organisms with continuous growth fecundity increases continually with age. Before maturation m_x will be zero, so the first age at which m_x is positive is also the age at maturation (note that in life history we define this age based on observed reproduction, not on physiological traits of maturity).

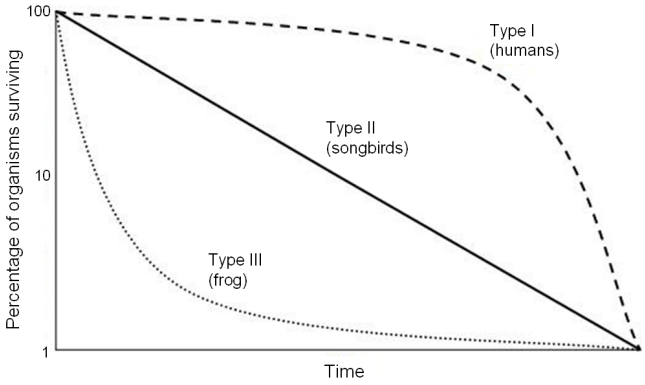


Figure 2.1: Three main types of survivorship curves (note the log scaled y-axis). Credit: Rayhusthwaite at English Wikipedia / CC-BY (https://creativecommons.org/licenses/by/3.0)

2.3 Life history strategies and trade-offs

Life history strategies refer to the different combinations of **life history traits** shown by species or populations, summarized in their age-specific survivorship l_x and fecundity m_x. The life history is a phenotype, and has a genetic and environmental component. There will always be **trade-offs** among the life history traits, as individuals have limited access to time and resources, and cannot invest in all traits simultaneously (if they could, evolution would lead to 'Darwinian demons' being immortal with infinite amount of offspring). At the species level, life history strategies are the combinations of life history traits that optimize **fitness** for a given environment. There is more than one solution (strategy) to this ''optimization problem", which is reflected in the wide range of life histories shown across different species.

Evolutionary changes require the combination of a selection pressure and genetic variation for the traits that are under selection. Demographic methods allow us to quantify selection pressures (through sensitivities, as we will see later) - other methods are needed to quantify the genetic variation of the traits. But before we can define selection pressures, we need a relevant measure of fitness, and we need to understand how fitness depends on the life history (l_x and m_x). In the next sections we will go through theory and methods providing this important link, as well as methods to calculate key life history parameters and selection pressures.

2.4 A basic life table

In its most basic form, a life table is a list of age-specific parameters of survival (survivorship l_x, and annual survival probability p_x) and fecundity (m_x):

Age x	Survival probability p_x	Survivorship l_x	Fecundity m_x
0	p_0	$l_0 = 1$	m_0
1	p_1	$l_1 = p_0$	m_1
2	p_2	$l_2 = l_1 p_1$	m_2
⋮	⋮	⋮	⋮
k	$p_k = 0$	$l_k = l_{k-1} p_{k-1}$	m_k

No individual lives past the final age, so the annual survival probability in the final age class is $p_k = 0$. However, before they die, individuals in this final class can produce m_k offspring on their k'th birthday. The fecundity will also be zero for the first age classes, until the age of maturation (by definition the first age at which offspring are reproduced).

For sexual species, life tables are often defined only for the female subpopulation, under the assumption that survival and reproduction of females are not influenced by the males. If this assumption holds, the male and female subpopulation will grow with the same rate λ, and it does not matter which of the subpopulations we use to describe population growth. However, if female reproduction or survival is limited by the number of males, more complex two-sex models are needed to describe the population growth [see e.g. Caswell, 2001]. We will not consider such models here.

Assuming the life table applies to females only, the fecundity parameter m_x represents the number of female offspring per female, unless it is specifically mentioned that it includes both male and female offspring. In that case we still need to find the number of female offspring, which is the relevant number for the life table. This depends on the offspring sex ratio. If the offspring sex ratio is 1:1, the number of female offspring is simply the number of offspring divided by 2.

2.4.1 Songbird example

As an example, consider a songbird that can live up to 3 years, then reproduces for the last time and dies before reaching age 4. The life table for the female population is given by

The final column of this life table lists the product of survivorship and fecundity, $l_x m_x$, and represents the expected number of offspring that an age 0 individual

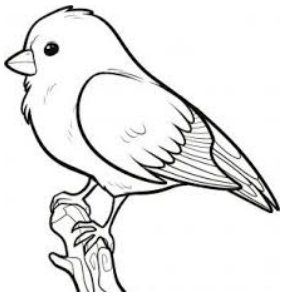


Figure 2.2: A songbird.

Table 2.2: Life table for the bird example.

Age x	mx	lx	px	lxmx
0	0	1.000	0.2	0.000
1	0	0.200	0.9	0.000
2	3	0.180	0.6	0.540
3	6	0.108	0.0	0.648

will produce at each age through life. This product occurs in the calculations of R_0, T_C and λ, and it is therefore often useful to include a separate column for it. We will get back to this example throughout this and the next chapter, and some of the R code builds on previous code for this example. Collected parameters and results for this example are provided in appendix B. Copy paste this if you want to run later code chunks for the bird example alone without running all the previous code chunks (you still need to define the relevant R functions).

To produce this life table in R, and plot the columns, we can use the following code:

```
x <- 0:3 #Age
k <- length(x)
px <- c(.2, .9, .6, 0) #Annual survival probability
lx <- c(1, cumprod(px)[-k]) #survivorship
mx <- c(0, 0, 3, 6) #Fecundity

songbird.lifetable <- data.frame("x" = x, "lx"=lx, "px"=px,
                                 "mx"=mx, "lxmx"=lx*mx)

#For plotting it is useful to use the long format:
songbirds.long <- songbird.lifetable %>%
  pivot_longer(c(lx, px,  mx,lxmx),
               names_to = "VitalRates", values_to = "Value")
```

```
songbirds.long$VitalRates <- factor( songbirds.long$VitalRates,
                    levels=c("px", "lx" , "mx", "lxmx"))

ggplot(songbirds.long) +
 geom_col(aes(x=x,y=Value, fill=VitalRates), lwd=1.2)+
 facet_wrap(~VitalRates, scales="free")+
 scale_fill_manual(values=colors4)+
 theme_bw() +
 labs( x="Age (year)", y="Value")+
  theme(legend.position = "none")
```

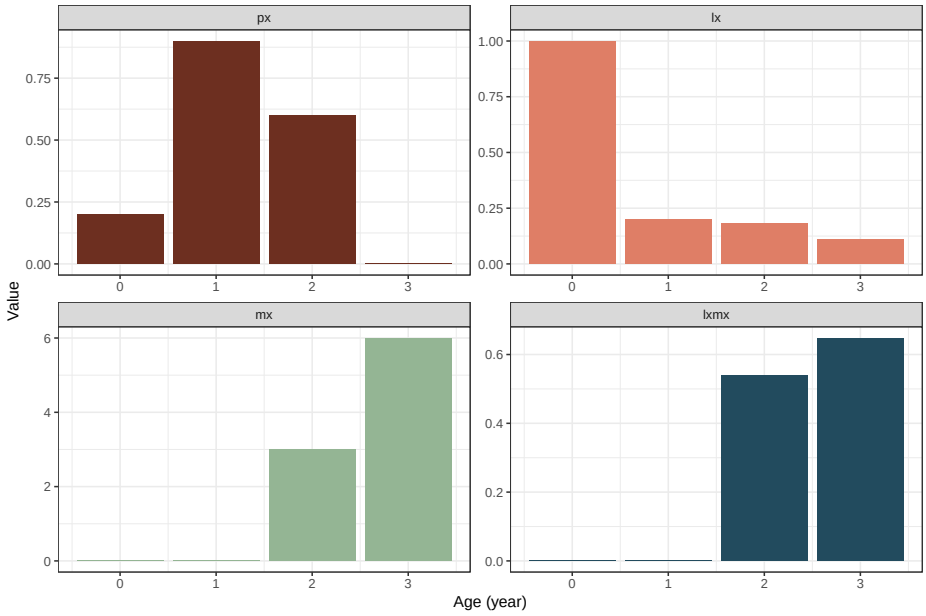


Figure 2.3: Plot of columns from the songbird life table (table 2.2): Age-specific survivorship l_x, annual survival p_x, fecundity m_x, and the product $l_x m_x$.

2.5 Dynamic and static life tables

There are two main ways of collecting data and estimating the age-specific parameters of survival and reproduction in a population. With a *dynamic life table*, also known as a cohort life table, an entire cohort of individuals are followed from birth to death, and their survivorship and fecundity is recorded at each age. This approach is often expensive and time-demanding, and in many cases it is not practically possible to obtain such data. The other main approach is the *static life table*, which is based on recording the survival and reproduction

of individuals from the whole population (all age groups) over one or a few time steps. This approach is usually more feasible, and corresponds to taking a cross-section of the population at a given time. However, the approach assumes that we are able to determine the age of individuals, which is difficult in many species. In many studies, more coarse age classifications such as "juvenile" or "adult" are used.

Importantly, the static life table approach is based on the assumptions that 1) the population is stationary (constant or near constant population size over time), and 2) the population has a stable age structure (the proportions of individuals in each age class do not change over time). A dynamic and static life table for a given population will be the same if the age-specific parameters of survival and fecundity are constant over time. For real populations this is rarely the case, but the life tables can still provide a good approximation.

Empirical data in demography

To get reliable empirical estimates of mean survival and fecundity we need long-term individual-based data. Incorporating effects of climate change and other external changes requires even longer time series, as some environmental events occur only rarely but can have large impact on the population.

Study systems providing such detailed long-term data from wild populations are rare, and require much effort to maintain. Those we have are extremely valuable, and much of our ecological and evolutionary knowledge is derived from these study systems.

2.6 Life table calculations

Once we have a life table with columns representing age, survivorship and fecundity, several useful quantities can be calculated. We will focus on three of them: The net reproductive rate R_0, the cohort generation time T_C, and the long-term growth rate λ.

All of these quantities can also be calculated directly from a projection matrix, as we will see later (chapter @ref(#s03-MatrixModels)).

2.6.1 Net reproductive rate

The **net reproductive rate** represents the number of female offspring by which each female will be replaced over her lifetime, on average, and is also known as the lifetime reproductive sucess:

$$R_0 = \sum_{x=0}^{k} l_x m_x. \tag{2.2}$$

The product $l_x m_x$ represents the number of offspring produced at age x given that the individual survives to that age. Thus, summing up this product for all ages we get the net reproductive rate.

The net reproductive rate is a measure of the population growth rate per generation. For instance, if $R_0 = 1.1$ the population will grow by 10% each generation. Thus, R_0 holds information on the population growth rate λ, although the two are not the same (λ is per time step, while R_0 is per generation). If $R_0 > 1$, we know that the population is growing ($\lambda > 1$), if $R_0 = 1$ then $\lambda = 1$ and the population is stable, and if $R_0 < 1$ then $\lambda < 1$, and the population is declining.

Bird example

From the bird life table (table 2.2), we see that the age at first reproduction is 2, and that although few individuals live to age 3, this is the age class of highest expected number of offspring. The net reproductive rate is in this case

$$R_0 = \sum_{x=0}^{3} l_x m_x = 0 + 0 + 0.54 + 0.648 = 1.188.$$

Thus, each female is expected to be replaced by 1.188 female offspring on average. From this number we now know that the population is growing ($\lambda > 1$ since $R_0 > 1$), although we do not yet know the value of the population growth rate λ.

In R, we can find this from the life table (or the vectors) as follows:

```
R0.lifetable <- function(lx, mx){
  sum(lx*mx)
}

#Bird example:
R0_Bird <- R0.lifetable(lx, mx)
R0_Bird
```

```
## [1] 1.188
```

Note that R automatically multiplies each element of the vectors first, before summing the products.

2.6.2 Cohort generation time

The generation time reflects the 'pace of life' of a species. Many biological properties (physiological rates, other life history parameters) scale with generation time. For age structured populations with overlapping generations there are different ways to define generation time. We will use the socalled **cohort generation time**, which is the mean age at reproduction of all the R_0 offspring the average individual produces:

$$T_C = \frac{1}{R_0} \sum_{x=0}^{k} x l_x m_x. \tag{2.3}$$

The expression $x l_x m_x / R_0$ represents the fraction of the mean lifetime reproduction (R_0) that happens at age x. By summing this up for all ages (and moving $1/R_0$ outside the sum) we get the average age at which reproduction occurs.

Bird example

Using the measure defined above and the values in table 2.2, the cohort generation time for the songbird is given by

$$T_C = \frac{1}{R_0} \sum_{x=0}^{k} x l_x m_x = \frac{1}{1.188}(0 + 0 + 2 \cdot 0.54 + 3 \cdot 0.648) = 2.54.$$

In R, we can find this as follows:

```
TC.lifetable <- function(x, lx, mx){
  R0 <- R0.lifetable(lx=lx, mx=mx)
  sum(x*lx*mx)/R0_Bird
}

#Bird example:

TC_Bird <- TC.lifetable(x,  lx, mx)

TC_Bird
```

```
## [1] 2.545455
```

2.6.3 Approximation of long-term growth rate

Since R_0 is an estimate of the per generation growth rate and T_C an estimate of generation time, we can use them to estimate the population growth rate per

time step, the intrinsic growth rate r (see chapter 1 on exponential growth):

$$\ln \lambda = r \approx \frac{\ln R_0}{T_C}. \tag{2.4}$$

This approximation assumes that all offspring produced during the lifetime on average (R_0) are produced exactly at age T_C, rather than at different ages. It can be a good starting point for the more accurate calculation of λ based on the Euler-Lotka equation, which is the topic of the next section.

Bird example

Using the values calculated above, we get the following approximation of λ for the songbird:

The estimate of λ based on R_0 and T_C is given by

$$\lambda \approx e^{\ln R_0/T_C} = e^{\ln 1.188/2.54} \approx 1.07.$$

In R, we can find this as follows, using the bird example:

```
exp(log(R0_Bird)/TC_Bird)
```

```
## [1] 1.070021
```

2.6.4 Euler-lotka equation

The **Euler-Lotka equation** is an essential equation in life history theory, because it provides the link between the life history of the focal population, as described by the age-specific fecundity m_x and survivorship l_x, and the mean fitness λ (long-term population growth rate), associated with that life history.

To derive the Euler-Lotka equation, we first define the number of newborn individuals at time t, which is the sum of the newborns produced at each age:

$$n_0(t) = \sum_{x=0}^{k} n_0(t-x) l_x m_x. \tag{2.5}$$

Here, $n_0(t-x)$ is the number of newborn x years ago (that are now aged x years). This number is multiplied by $l(x)$ to obtain the number of them that are still alive at age x, and then by $m(x)$ to get the total number of offspring they produce at age x. The sum is over all ages, to get the total number of offspring (left side of the equation). For the next step of the derivation, we

first assume that the population, and therefore each age class, grows steadily and exponentially at rate λ. Because the population grows exponentially, we assume that the number of newborn over time will follow an equation on the form $n_0(t) = Q\lambda^t$ (remember the equation for exponential growth), where Q is an unknown constant. Inserting this for $n_0(t)$ and $n_0(t-x)$ in the equation above, we now get

$$Q\lambda^t = \sum_{x=0}^{k} Q\lambda^{t-x} l_x m_x. \tag{2.6}$$

Finally, we divide each side by $Q\lambda^t$ to get the Euler-Lotka equation (remember that we can write $\lambda^{t-x} = \lambda^t \lambda^{-x}$):

$$1 = \sum_{x=0}^{k} \lambda^{-x} l_x m_x. \tag{2.7}$$

It is now tempting to try to rearrange this equation to isolate λ on one side, to get a 'nice' expression for λ as a function of the cumulative survival and fecundity. Unfortunately, this is not possible (except for a few special cases)! This equation can only be solved numerically, for instance by trial and error (change the value of λ in the expression until you find the one that leads to the sum being close to 1).

Bird example

The exact value of λ is defined by the Euler-Lotka equation, which for the bird example (table2.2) is given by

$$1 = \sum_{x=0}^{k} \lambda^{-x} l_x m_x = 0.54\lambda^{-2} + 0.648\lambda^{-3}.$$

We can solve this by trial and error, and a good starting value for λ is the estimate from R_0 and T_C, $\lambda \approx 1.07$. However, solving the equation is more easily done in R.

One of several possible approaches is to use the function `uniroot()`. You are not expected to memorize the details of this approach, but it is shown here as it can be useful with lifetables. Uniroot searches for the solution of a function (i.e. the value of its first argument that makes the expression in the function equal to zero) within a predefined range. For this to work for the Euler-Lotka equation, we need to rearrange it to get an expression that equals zero (just move the number 1 from right-hand side to left hand side), make sure λ is the first argument of this function, and then apply the uniroot function:

```r
#Rearranging  Euler-Lotka function:
EulerLotka <- function(lambda, x, lx, mx){
  sum(lambda^(-x)*lx*mx)-1
}

#Apply uniroot to find lambda, store the result
lambdabird <- uniroot(EulerLotka, x=x,
    lx=lx, mx=mx, interval=c(.1,2))$root
```

With this approach the estimated value is $\lambda = 1.07025$, close to the approximated value calculated above.

You have now seen an example of how the Euler-Lotka equation can be solved in R, but this approach is not commonly used. It is much more convenient to find λ based on the projection matrix from a matrix model, which is the topic of section 3.

2.7 Fitness measures in life history theory

In the context of life history theory, fitness is often measured by the long-term growth rate λ (or equivalently $r = \ln \lambda$) as defined by the Euler-Lotka equation, or by the dominant eigenvalue of the projection matrix (chapter 3).

Another fitness measure which is commonly used, is the lifetime reproductive success, R_0 (equation (2.2)). This measure does however not account for timing of life history events (at which ages reproduction happens, for instance), and thus ignores an important aspect of the life history. Both λ and R_0 are based on the assumption of constant age-specific survival and fecundity in the life table, and neither is a good measure of fitness in the case of strong density- or frequency-dependence (because λ and R_0 are both based on average values that are not useful for understanding selection in such situations). Life history theory deals mostly with outcomes of evolutionary processes in the long run (i.e. evolution of entire life histories) and to a less extent the short-term dynamics of gene frequencies (population genetics). With strong density or frequency dependence, and where the purpose is to estimate gene frequencies over time, other methods are better suited (e.g. adaptive dynamics frameworks). These are not considered here, but note that the measure of fitness will also generally be different in these cases.

Following the convention through most of the literature on life history evolution and quantitative genetics based on demographic models, λ is the main fitness measure for a structured population in a constant environments.

2.8 Semelparity and iteroparity

Some species, including most mammals, have multiple reproductive events during their lifetime. This is known as **iteroparity**, while **semelparity** applies to species that reproduce only once. This mode of reproduction is used by annual plants as well as some perennials, several invertebrates, many fish species, and a few mammals. In salmon, some species are iteroparous and others are semelparous. From a life history perspective it is interesting to know under which circumstances evolution of iteroparity versus semelparity is favored, and we can use the knowledge from life table methods to investigate this question.

Cole [1954] was among the authors who were interested in this and other life history questions. He is known for **Cole's paradox**, which was only a small part of a larger article. In this example, Cole compared two species: The first is semelparous and lives for only a year, after which it produces m_1 offspring and dies. The second is iteroparous and lives forever (survival probability of 1 each year), producing m_2 offspring each year. Now look at the annual growth rate (fitness) λ for each species: Species 1 gets $\lambda_1 = m_1$ (since each individual leaves behind m_1 individuals on average each year), while the second species gets $\lambda_2 = 1 + m_2$ (each individual leaves behind $1 + m_2$ individuals each year, i.e. itself plus offspring). What would it take for the semelparous species to have higher fitness than the iteroparous one?

$$\lambda_1 > \lambda_2 \tag{2.8}$$

$$m_1 > 1 + m_2. \tag{2.9}$$

In other words, as long as the semelparous species (species 1) produces at least 1 more offspring per clutch than the iteroparous one (species 2), it would have the same or higher fitness. This is known as Cole's paradox, as it seems unlikely that an immortal species should get the same fitness as a mortal one having just one more offspring per clutch. It is also a paradox that semelparity is not more common (in birds an mammals) if semelparity is so 'cheap' to evolve. The clue to 'resolve' the paradox is that the assumptions for each species are too simplistic to apply to the real world. Once we take more realism into account, for instance drop immortality but instead have an annual survival probability $p < 1$ for the iteroparous species, and introduce a difference between adults and juveniles (where juveniles cannot reproduce), iteroparity becomes relatively more favorable and it would cost more to switch to a semelparous life history.

Note that while semelparity is 'rare' in birds and mammals, which likely motivated the need to explain how it evolved (compared to iteroparity), it is common in other taxa.

2.9 The Lack clutch

The concept of the **Lack clutch** originates from ornithology, but applies generally to other organisms as well. It is named after the ornithologist David Lack, who suggested a pioneering baseline model for understanding life history trade-offs that affect clutch size [Lack, 1947].

The problem started from the observation, including brood manipulation studies, that many birds seem to lay fewer eggs per clutch than they could manage to raise to fledging (independence). This suggests there could be one or more life history trade-offs constraining birds from laying larger clutches, and that the observed clutch size perhaps represents some fitness optimum for the current environment. Lack's model assumes that this trade-off is with fledgling survival, and he also assumed this trade off was linear. In that case, the optimum clutch size, from the parent's point of view, is the one that maximizes the number of offspring that survive to fledging. This clutch size is known as the Lack clutch. Each offspring, on the other hand, would prefer (from a fitness perspective) to have all parental effort to itself, so that the optimum clutch size from the offspring point of view is 1. This is an example of a **parent-offspring conflict** (where offspring and parents do not 'agree', evolutionarily speaking, on what is the optimum parental investment in a clutch).

We can describe the Lack clutch model mathematically as follows. Let N_e be the number of eggs laid in a clutch, and let $l_f(N_e)$ be the probability of survival from egg laying until fledging (when offspring are independent). It is a linear function of the number of eggs in the clutch:

$$l_f(N_e) = 1 - cN_e, \tag{2.10}$$

where c is the slope of the relationship. The number of survived fledglings is then given by

$$\begin{aligned} N_f(N_e) &= l_f(N_e)N_e && (2.11)\\ &= (1 - cN_e)N_e && (2.12)\\ &= N_e - cN_e^2. && (2.13) \end{aligned}$$

This is a quadratic function, and we find the maximum (or minimum) where the derivative is equal to zero. The corresponding value of N_e is the Lack clutch N_e^*:

$$\frac{dN_f(N_e)}{dN_e} = 1 - 2cN_e \tag{2.14}$$

$$1 - 2cN_e^* = 0 \tag{2.15}$$

$$N_e^* = \frac{1}{2c}. \tag{2.16}$$

This tells us that the stronger the trade-off with offspring survival (size of the slope c), the lower the Lack clutch size. The Lack clutch is larger than one only for $c < 0.5$.

The Lack clutch is based on a simple model, with some important assumptions that do not necessarily hold for real world examples. The model considers only one (but important) trade-off: Clutch size versus offspring survival until fledging. However, other trade-offs could also affect the clutch size, such as parental survival and future reproduction. Another important assumption is that the trade-off between offspring survival and clutch size is linear, but this is just a convenient assumption. In many cases the offspring survival could be a non-linear function of clutch size (for instance if parasitism affects large clutches more than small ones). Furthermore, the model assumes a constant environment, and that parents treat all offspring of the clutch equally. In many species, for instance in many raptors, parents favor the largest / oldest of the siblings, so that younger siblings only survive in very good conditions (or if the older ones die for some reason). In these cases, the younger offspring may represent an "insurance" for the parents, and effects like this are not captured in the simple Lack clutch model. Still, this is a good baseline model for exploration of more complex hypotheses around the evolution of clutch size.

2.10 Exercises

For suggested solutions, see appendix A.

Exercise 2.1

The table shows the average lifetime number of children per woman in some different countries.

Assume exponential growth, and use the approximation in equation (2.4) to answer the questions:

1. What is the annual population growth rate (λ) of each country, given different assumptions for the value of generation time?

Table 2.3: Average lifetime number of children in different countries (2019 data from http://worldpopulationreview.com/countries/total-fertility-rate/).

Country	Children per woman
Mali	5.8
Tunisia	2.2
Australia	1.7
Norway	1.6
South Korea	0.9

2. How many years does each country need to double in population size, if we assume a generation time of 25 years?

3. How well / poorly do you think these populations meet the assumptions made for these calculations?

Exercise 2.2

Figure 2.4: Flowering plant

A botanist follows a cohort of a flowering plant from seeds (age 0) until all are dead. He marks the seedlings and comes back every year and counts how many living plants are remaining in the cohort. When they start to reproduce he counts the number of seeds per plant. Based on this information he writes the following incomplete life table:

1. What is the age at first reproduction of this plant?

2. Create the same life table in R, with columns corresponding to Age x, Remaining cohort n_x, and fecundity m_x.

Table 2.4: Incomplete life table for a flowering plant.

Age x	Remaining cohort n_x	Fecundity m_x
0	12376	0.0
1	4233	0.0
2	1790	0.0
3	340	2.1
4	268	3.2

3. Calculate the vectors of cumulative survival probability l_x and annual survival probability p_x, and add these as columns to the life table you created.
4. Make plots of each life table column (n_x, l_x, p_x, m_x) as a function of age x. Describe the life history you see from the plot.
5. Calculate the mean lifetime reproduction R_0 from the life table. What does this value say about the population growth rate?
6. Calculate the generation time T_C from the life table. What does this measure actually describe? What is the estimate of λ based on R_0 and T_C?
7. Solve the Euler-Lotka equation for λ using the method demonstrated for the bird example. How does this value of λ differ from the estimate based on R_0 and T_C?

Exercise 2.3

1. In the bird example (section 2.4.1), the values of fecundity given in table 2.2 correspond to the mean number of fledged female offspring per adult female. Assume that the offspring survival (from eggs to fledging) is 0.9, what is the average clutch size (number of eggs laid) at each age?
2. If we assume that the assumptions of the Lack clutch apply, what is the trade-off between offspring survival and offspring number at each age?
3. In another bird species, a close relative to this example bird, individuals only reproduce at age 3. Otherwise this species has the same average fitness and the same survivorship as in our bird example. What is the fecundity in this other species?
4. What is the name of the reproductive mode of the two bird species (the example bird and its relative)? Discuss why (or why not) the two species fit with the result of Cole's paradox.

Chapter 3

Age-structured matrix population models

Matrix models make it possible to work with systems of multiple equations simultaneously, which is what we need to describe the growth of structured populations. For instance, if we study a long-lived species with 50 age classes, it is very cumbersome to write down the growth equations for all of the 50 age classes. With a matrix model we need only one equation to keep track of the growth of all of stages simultaneously.

This chapter will deal with age-structured matrix models and introduce methods for calculating key properties of life history and population growth. Chapter @ref{s04-stage} will introduce more general stage structured models, and we will see that the methods introduced here for age-structured populations are applicable also to these cases.

The main 'ingredient' of an age-structured matrix model is the Leslie matrix, which contains all the information necessary to project the growth of each age class over time. Several life history parameters can also be calculated directly from this matrix, including the long-term population growth rate λ.

3.1 Learning goals

- Define a Leslie matrix for age-structured populations, and its two main components.
- Explain the differences between a pre- and post-reproductive census.
- Show how a Leslie matrix is related to the life table.
- Draw a life cycle graph and explain how it is related to the Leslie matrix.

- Calculate asymptotic parameters from a matrix model (using projection or eigen analysis), and explain their biological meaning.
- Calculate lifetime reproduction, generation time and life expectancy from the Leslie matrix and its main components.
- Calculate sensitivities and elasticities of λ with respect to projection matrix elements or underlying parameters.
- Do an LTRE analysis to understand the difference in the long-term growth rate in two different environments.

3.2 Age-structured growth

In this section we will define the growth of each age class as a function of the annual survival probability and fertility, and introduce the matrix equation to describe this growth. In the next sections we will see how the age-structured matrix model is connected to life tables, and how it can be described by a life cycle graph. The last sections present important calculations that can be done with a matrix model.

We will denote the first class as class 1, the second as class 2, etc., but as we will see in section 3.3, the definition of the parameters of survival and reproduction depend on census time (when the population is counted within each time step). To keep track of the number of individuals in each of the age classes over time, we define the following **population vector $\mathbf{n}_t$** of individuals in age class 1 to k:

$$\mathbf{n}_t = \begin{bmatrix} n_1(t) \\ n_2(t) \\ \vdots \\ n_k(t) \end{bmatrix}. \tag{3.1}$$

The total population size at time t is the sum of this vector, $N_t = \sum_{j=1}^{k} n_j(t)$ (note that we now use j or i as index instead of x, following conventional notation for matrix models). Let s_j describe the survival probability of individuals in age class j to the next time step, and f_j the number of offspring produced by each individual in age class j that enter age class 1 at time $t+1$. Importantly, s_j and f_j of the Leslie matrix are *not the same* as the age-specific survival probability p_j and fecundity m_j (see section 3.3).

The growth of age class 1 (offspring) from one time step to the next is then given by

$$n_1(t+1) = f_1 n_1(t) + f_2 n_2(t) + ... + f_k n_k(t), \tag{3.2}$$

while the growth of the other age classes is given by

$$\begin{aligned} n_2(t+1) &= s_1 n_1(t), \\ n_3(t+1) &= s_2 n_2(t), \\ &\vdots \\ n_k(t+1) &= s_{k-1} n_{k-1}(t). \end{aligned} \tag{3.3}$$

Together, theses equations constitute a system of linear equations, which we can describe with a matrix model:

$$\begin{aligned} \mathbf{n}_{t+1} &= \mathbf{A}\mathbf{n}_t \\ \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ \\ n_k \end{bmatrix}_{t+1} &= \begin{bmatrix} f_1 & f_2 & & \cdots & f_k \\ s_1 & 0 & & \cdots & 0 \\ 0 & s_2 & & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & s_{k-1} & 0 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ \\ n_k \end{bmatrix}_{t} . \end{aligned} \tag{3.4}$$

This is the age-structured version of the exponential growth model of an unstructured population (section 1). The matrix $\mathbf{A}$ is a square matrix (same number of rows and columns), of dimension $k \times k$, where k is the number of age classes. It is named a **Leslie matrix** after the work of Leslie [1945; 1948], and the more general term which also applies to other kinds of structure is a **projection matrix**. The elements of the first row of $\mathbf{A}$ represent the production of offspring to next year's population from each of the age classes. Here we refer to these elements as **fertility coefficients** or simply fertilities, while the **survival coefficients** are located along the sub-diagonal of the Leslie matrix. In the final age class, no individuals survive until the next time step, but individuals can reproduce and contribute with a number f_k to next year's age class 1.

The Leslie matrix can be decomposed as the sum of a **survival matrix U** and a **fertility matrix F**:

$$\begin{aligned} \mathbf{A} &= \mathbf{U} + \mathbf{F} \\ \begin{bmatrix} f_1 & f_2 & & \cdots & f_k \\ s_1 & 0 & & \cdots & 0 \\ 0 & s_2 & & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & s_{k-1} & 0 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & & \cdots & 0 \\ s_1 & 0 & & \cdots & 0 \\ 0 & s_2 & & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & s_{k-1} & 0 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & & \cdots & f_k \\ 0 & 0 & & \cdots & 0 \\ 0 & 0 & & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.5}$$

These two sub-matrices are used in the calculations of the net reproductive rate R_0 and generation time G (section 3.6). In chapter 4 we will further decompose these two matrices into components describing survival, fertility and stage transitions.

3.2.1 Creating a Leslie matrix in R

The following R function creates a Leslie matrix from vectors of survival **s** and fertility **f** (the vectors must have the same length, corresponding to the number of age classes). The final element of the survival vector should be 0 if the model is purely age-structured (otherwise, individuals are allowed to accumulate in the final class, which is a commonly used extension of age structured models).

```
Create.Amat <- function(Svec, Fvec){
  k <- length(Svec)
  MatA<- matrix(0,k,k)
  MatA[1,] <- Fvec
  for(i in 1:(k-1)){
    MatA[i+1,i] <- Svec[i]
  }
  #Individuals can accumulate if Svec[k]>0:
  MatA[k,k] <- Svec[k]
  MatA
  }
```

3.3 Pre- and post-reproductive census

The Leslie matrix depends on coefficients of fertility, f_j, and survival probabilities s_j. How these coefficients are linked to the life table depends on the *census type* of the matrix model, in other words at what time point the population is counted during each time step For a birth-pulse population (where individuals reproduce once per time step), there are two main census types: Pre- and post-reproductive* census. As the names indicate, they assume that the population is counted either right before (pre) or right after (post) reproduction.

Census time is important because a lot can happen to a population also within the time intervals of the model. The age-specific parameters describing fecundity (m_x) and survival (l_x, p_x) as a function of actual age x are the same regardless of census time. But the fertility and survival coefficients of the Leslie matrix (s_j, f_j) will be different depending on census - see definitions below.

In general, the size of age class 1 is larger in the post-reproductive census than with pre-reproductive, because individuals in this class are newborn. For any age-structured population with a positive growth rate, the stable age structure (proportion of individuals of each age) will always decline with age.

We assume that the population reproduces only once per year. But different species can have very different modes of reproduction, depending on their life history and environments. Some tropical species can reproduce at all times through the year. Bacteria also reproduce (divide) continuously. Other species can have more than one reproductive events each season, or reproduce only every other season (or even more rarely). It is possible to account for these different types of reproductive mode in a matrix model [Caswell, 2001], but that requires a different way to estimate the age-specific survival and fertility coefficients, which is beyond the scope of this course.

Below we go through the parameters of a pre- and post-reproductive census in more detail, using the bird example.

3.3.1 Pre-reproductive census

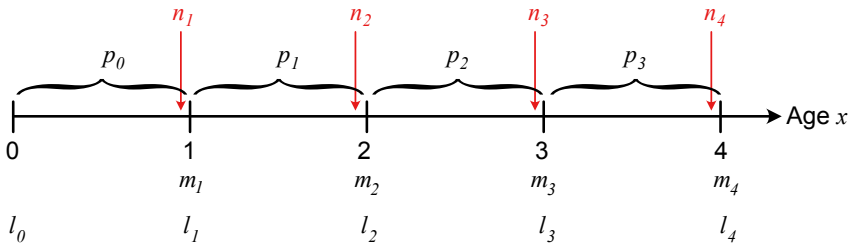


Figure 3.1: Overview of the timing of different events of survival and reproduction within each time step, for a pre-reproductive census.

With a **pre-reproductive census**, offspring will be nearly 1 year old when counted, and age classes correspond to the actual age at counting (e.g. individuals in age class 1 are 1 year old when counted), so that $s_j = p_j$. The fertility coefficients are then given by $f_j = p_0 m_j$, the product of the fecundity m_j and the first year survival probability p_0.

Bird example

Using the bird example introduced in section 2.4.1, the Leslie matrix for a pre-reproductive census is given by

$$\mathbf{A} = \begin{bmatrix} f_1 & f_2 & f_3 \\ s_1 & 0 & 0 \\ 0 & s_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & p_0 m_2 & p_0 m_3 \\ p_1 & 0 & 0 \\ 0 & p_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.6 & 1.2 \\ 0.9 & 0 & 0 \\ 0 & 0.6 & 0 \end{bmatrix}. \qquad (3.6)$$

The classes of this matrix corresponds to age 1, 2, and 3. We do not 'see' the 0-year olds in the matrix model, but the survival probability p_0 enters the fertility coefficients.

In R, we can define the matrix as follows:

```
Abird.pre <- Create.Amat(Svec <- c(.9, .6, 0), Fvec= c(0, .6, 1.2))
```

3.3.2 Post-reproductive census

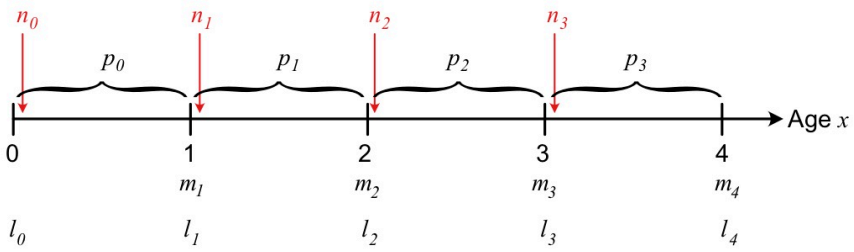


Figure 3.2: Overview of the timing of different events of survival and reproduction within each time step, for a post-reproductive census.

With a **post-reproductive census**, the population is counted right after reproduction. In this case, age class 1 consists of newborn individuals (age 0), and individuals in age class j have the actual age $j-1$, so that $s_j = p_{j-1}$. The fertility coefficient f_j now depends on survival of the parent from last time step s_{j-1}, and on fecundity m_j of the age j, since individuals reproduce right before next census. Thus, the fertility coefficient of the Leslie matrix is now $f_j = p_{j-1}m_j$. This definition of fertilities can be counter-intuitive, as the contribution from a given age class j to next year's offspring depends on survival from the previous (actual) age p_{j-1}.

Bird example

For the bird example, we get the following Leslie matrix with post-reproductive census:

$$\mathbf{A} = \begin{bmatrix} f_1 & f_2 & f_3 \\ s_1 & 0 & 0 \\ 0 & s_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & p_1 m_2 & p_2 m_3 \\ p_0 & 0 & 0 \\ 0 & p_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2.7 & 3.6 \\ 0.2 & 0 & 0 \\ 0 & 0.9 & 0 \end{bmatrix}. \tag{3.7}$$

Note that each class of this matrix corresponds to age 0, 1, and 2. Thus, in this case we do not 'see' the 3-year olds as a separate class, but their fecundity defines fertility coefficient in the third class.

In R, we can define this matrix as follows:

```
Abird.post <- Create.Amat(Svec <- c(.2, .9, 0), Fvec= c(0, 2.7, 3.6))
```

3.3.3 Avoiding mistakes

It turns out that mistakes in constructing the projection matrix are quite common in the ecological literature [Kendall et al., 2019], and issues with census time often play a role here. Mistakes are easily made, since keeping track of age classes versus actual age requires accurate 'book-keeping', especially with a post-reproductive census. Mistakes are also more common for post-reproductive census models than pre-reproductive census models [Kendall et al., 2019].

Contributing to the confusion, the terminology of matrix models is not consistent in the literature. For instance, what we have defined as the 'fertility coefficients' f_j is by some authors referred to as 'fecundity' (m_j in our notation). Therefore, some authors mistakenly did not include survival in their fertility coefficients. Finally, there is no general consensus on whether to use 0 or 1 as index for the first age class when a post-reproductive census is used (note that we always use 1 as the index, regardless of census time), and this choice can affect the limits and indexing in different calculations (e.g. the Euler-Lotka equation).

The main recommendation to avoid such errors when constructing a matrix model, is to take your time and make sure you have defined each element of the matrix model and its biological meaning. One way to double check that you have obtained the right coefficients is to compare a pre- and post-reproductive census model and Leslie matrix, and check that λ calculated from the matrix (section3.6) is the same for the two cases. If it differs, at least one of the models has an error. It can also be helpful to draw a life cycle graph (see section 3.4).

The long-term population growth rate λ should be the same regardless of census time, and for age structured models the same is true for R_0 and generation time G (section3.6). But other properties will change between census times, such as the stable stage structure and reproductive value. The difference can be large, since many changes can occur within time steps. This is particularly true for species producing a many offspring per reproductive event, with low offspring survival. A female cod, for instance, can produce millions of eggs, but only a few will survive the first year. If we count a cod population right after spawning (post-reproductive census), the fertilized eggs would constitute a very high proportion of the population (practically the entire population). If we instead counted the population right before spawning (pre-reproductive census), the proportion of age 1 offspring would not be as high.

3.4 Life cycle graph

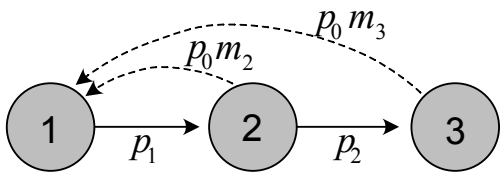


Figure 3.3: Life cycle graph for the songbird example, with a pre-reproductive census.

A structured model can be represented in different ways. You have already seen two examples for age-structured populations: A life table and a matrix model. A third representation, which is a helpful visual tool, is the life cycle graph. This is particularly useful for more complex structures than age structure, with many possible transitions among stages. For age structured popualtoins it can be a useful tool to keep track of parameters for the different census times.

A life cycle graph gives an overview of the possible transitions among age classes each time step. For age structured populations, these transitions are quite simple, as they only consist of of survival from one age class to the next, and reproductive contributions from each age class to next year's offspring.

Figure 3.3 shows the life cycle graph for the bird example introduced in section 2.4.1, for a pre-reproductive census. Each circle represent one age class of the matrix model, and since this is pre-reproductive census age class 1 consists of individuals at age 1, etc. The circles are connected by arrows representing survival and reproduction. Comparing the life cycle graph to the Leslie matrix described before (equation (3.6)), we see how the transitions are represented in the projection matrix.

3.5 Projected growth

The simplest application of a projection matrix is to use it to project the growth of each age class over time, from some given initial population vector $\mathbf{n}_0$ to a desired number of time steps T_{max}. The following R function performs such a projection for a given Leslie matrix $\mathbf{A}$, and returns the number of individuals in each age class at each time step in a data frame.

```
projection <- function(MatA, Tmax = 50, n0){
  k <- dim(MatA)[1]
  if(length(n0)!=k){
      warning("n0 should have length k
              corresponding to number
```

```
              of stages in Amat")
      }
  Nmat <- matrix(NA, nrow= Tmax+1, ncol=k)
  cnames <- paste("n", 1:k, sep="")
  timesteps <- 0:Tmax
  Nmat[1,] = n0
  for(i in 2:(Tmax+1)){
    Nmat[i,] = MatA%*%Nmat[i-1,]
  }
  frame <- data.frame(timesteps, Nmat)
  colnames(frame) <- c("time", cnames)
  frame
}
```

In R, the operator $\%*\%$ performs a matrix multiplication. Here this is done for each time step within a for-loop to calculate the current population vector as a function of the previous one. Note that the initial population vector n_0 has to have correct number of element k, i.e. the same number of age classes as in the projection matrix $\mathbf{A}$. If not you will get an error when using the function. The initial population vector is included the returned data frame (the first row).

Bird example

For the bird example (section 2.4.1), we can apply the function to project the population over time. The projection matrices for a pre- or post-reproductive census were defined earlier for this example. We can use the following code to project and plot the population over time for each census type:

```
#Return matrix of projected growth of each age class over time:
Pop.bird.pre <- projection(MatA=Abird.pre, n0=rep(10,3), Tmax=30)
Pop.bird.post <- projection(MatA=Abird.post, n0=rep(10,3),Tmax=30)

#Long format for plotting:
Pop.bird.pre.long <- Pop.bird.pre %>% pivot_longer(c(-time),
               names_to = "AgeClass", values_to = "Value")
Pop.bird.post.long <- Pop.bird.post %>% pivot_longer(c(-time),
               names_to = "AgeClass", values_to = "Value")

#Add census type and combine:
Pop.bird.pre.long$census <- "Pre"
Pop.bird.post.long$census <- "Post"
Pop.bird.long <- rbind(Pop.bird.pre.long,Pop.bird.post.long)

#Order levels of the census factor variables
Pop.bird.long$census <- factor(Pop.bird.long$census,
```

```
                                levels=c("Pre","Post"))

#Plot

ggplot(Pop.bird.long) +
  geom_line(aes(x=time,y=Value, col=AgeClass,
                linetype=AgeClass), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(census),ncol=2)+
  scale_color_manual(values=colors3)+ #Add colors manually
  scale_linetype_manual(values=c(1,2,3))+ #Add linetypes manually
  labs( x="Time (year)", y="Age class size")+ #Axis labels
  theme(legend.position = "top" ) #Place legend on top
```

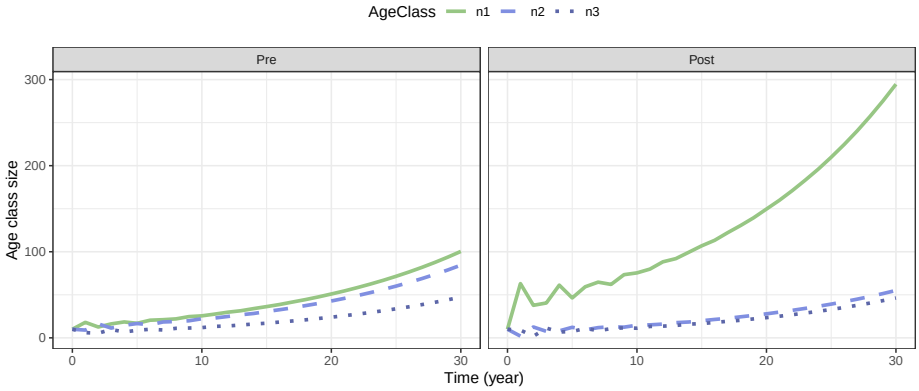


Figure 3.4: Projected growth of the bird example, with pre-reproductive (age class 1 consist of 1-year olds) or post-reproductive census (age class 1 consist of 0-year olds).

Figure 3.4 shows the difference between the pre-reproductive an post-reproductive census in the bird example. With pre-reproductive census the first age class consists of 1-year olds, the second class of 2-year olds and the third class of 3-year olds. With post-reproductive census the first class consists of individuals of age 0, the second of age 1 and the third of age 2. In each model we start with 10 individuals in each class, but the subsequent growth pattern is different because of how the population is counted. We see a higher number of individuals in the first class of the post reproductive census model compared to the other classes, due to high reproduction in this example.

Note the fluctuations in the beginning, that become smoother over time and eventually disappear. These occur because the population is started with 10 individuals in each class, which is not the stable structure of the population. The fluctuatoins are called **transient fluctuations** because they are temporary.

After some time they become negligible, as the population reachedes a phase of **stable growth**, where each age class grows exponentially with the same rate λ. In this phase the population shows **asymptotic dynamics**, and it will remain there unless something disturbs the stable structure. There are three properties of the population in the asymptotic phase: The long-term growth rate λ, the stable age structure $\mathbf{u}$, and the reproductive values $\mathbf{v}$. The next section describes how to calculate these properties from the projected growth, while section 3.6 shows how to calculate these properties from the projection matrix and its two main components. Here the different properties and their biological meaning are discussed in more detail.

3.6 Asymptotic properties

3.6.1 Calculations based on projected growth

From the population projections we can now estimate λ, the stable age structure $\mathbf{u}$, and the reproductive values $\mathbf{v}$, as shown in the next sections.

Long-term population growth rate λ:

This is the same growth rate λ that we have already defined based on the Euler-Lotka equation (section 2.6.4). This parameter can also be estimated from the projected population size over time, as $\lambda = N_{T+1}/N_T$ (where $N_t = \sum_{i=1}^{k} n_{i,t}$ is the total population size at time t), after a sufficiently large number of time steps T (large enough that the estimate of λ, to a chosen number of decimal places, does not change if we increase T further). It can also be estimated from any of the age classes,

For the bird example (here with prereproductive census), we can use the following R code:

```
Total.pop.bird <- apply(Pop.bird.pre[-1],1,sum) #Get total size
Tmax <- length(Total.pop.bird)#Recover max time
lambda.bird <- Total.pop.bird[Tmax]/Total.pop.bird[Tmax-1]
```

The estimated value from projection over 31 time steps is $\lambda \approx 1.07027$, close to the value $ $1.07025 calculated from the life table in section 2.6.4. The more time steps are used for the projection and calculation, the more accurate the estimate of λ will be.

Stable age structure:

The age structure is the proportion of the population constituted by each age class. As you can see from the projected growth (figure 3.4), the age structure

will fluctuate in the beginning, before it approaches a **stable age structure** as the transient fluctuations fade out, a structure that becomes constant over time. The stable age structure is denoted by the vector $\mathbf{u} = [u_1, u_2, ..., u_k]$ (where $\sum_{j=1}^{k} u_j = 1$). This vector can be estimated from the projected population, as $u_j \approx n_{j,T}/N_T$, where again T is a (sufficiently) large number of time steps. For growing age structured populations, the age structure will always be a monotonically decreasing function of age class.

In R, we can also estimate the stable structure from the projections, here shown for the bird example with pre-reproductive census:

```
stablestructure.pre <- Pop.bird.pre[Tmax,-1]/Total.pop.bird[Tmax]

k <- dim(Abird.pre)[1]

names(stablestructure.pre) <- paste("u",1:k,sep="")

stablestructure.pre
```

```
##           u1        u2        u3
## 31 0.4324629 0.3636527 0.2038843
```

Reproductive values:

We can calculate the age-specific **reproductive values** using a slightly modified version of projected growth. The reproductive value was first defined by Fisher [1930], who was interested in how much individuals of each age class would contribute to the future population, compared to other age classes. We can think of the reproductive value as a return rate associated with 'investing' in different age classes. If we could choose individuals from just one age class to start a new population, and wanted this population to be as large as possible after some time - we should choose the age class with the highest reproductive value.

To calculate the reproductive values from projections, we have two options: One is to project the total population size k times (where k the number of age classes), and at projection number i we start with a given number of individuals (e.g. 10) in age class i only (and zero in the other classes). The relative differences between the projected population sizes after T time steps then reflect the reproductive values. The vector $\mathbf{v}$ (a row vector) can be scaled in different ways, here we will use the scaling $\mathbf{vu} = \sum_{j=1}^{k} v_j u_j = 1$.

Another approach is to use the *transposed* Leslie matrix $\mathbf{A}^T$ (taking the transpose means 'flipping' the matrix over the main diagonal, so that rows and columns are switched) and do the same kind of projection as for the stable age structure before applying the scaling defined above.

We can use the following code to apply the second approach for the bird example (pre-reproductive model):

```
#Projected growth using the transpose Leslie matrix:
Pop.bird.transpose <- projection(MatA=t(Abird.pre), n0=rep(10,3))

k <- dim(Abird.pre)[1]

repvalues.pre <- Pop.bird.transpose[Tmax,-1]/
  sum(stablestructure.pre*Pop.bird.transpose[Tmax,-1])

names(repvalues.pre) <- paste("v",1:k,sep="")

repvalues.pre
```

```
##           v1       v2       v3
## 31 0.9144873 1.087482 1.025347
```

Note that the reproductive value of an age class is a relative measure (the value should always be interpreted in comparison to the other classes). If some environmental change alters the survival and fecundity in one age class, this will generally affect the reproductive value not only in this class, but in all the other age classes as well.

3.6.2 Calculations based on projection matrix

You have now seen how the Leslie matrix $\mathbf{A}$ (or its transpose) can be used to project the population over time, and how we could calculate λ, $\mathbf{u}$ and $\mathbf{v}$ based on these projections. This method of estimation is intuitive and works well for a given model, such as our bird example, but it takes some time and is not the most efficient way to calculate these properties. A much more efficient way is to calculate these properties directly from the projection matrix $\mathbf{A}$, using the eigenvalues and eigenvectors (socalled 'eigen analysis'.

The mathematical details of these calculations are outside the scope of this introduction (see e.g. Caswell [2001] for a more detailed introduction), but they are routinely included in R packages and easily defined in custom functions as well (as we will do below). In short, given that certain properties of $\mathbf{A}$ are fulfilled (which they are for most biologically relevant matrix models), λ is the dominant eigenvalue of $\mathbf{A}$, $\mathbf{u}$ is the right eigenvector associated with λ, and $\mathbf{v}$ is the left eigenvector. It is worth noting that for age-structured populations, the socalled *characteristic equation* of the matrix $\mathbf{A}$, from which we all of these parameters are calculated, corresponds to the Euler-Lotka equation we defined in section 2.6.4.

The following R function takes a Leslie matrix (or any other kind of projection matrix) as the input variable, and returns the value of λ along with the vectors

u and **v**, using the scaling defined above.

```
uvlambda <- function(MatA){
 ev <- eigen(MatA)
 tev <- eigen(t(MatA))
 lmax <- which.max(Re(ev$values))
 U <- ev$vectors
 V <- tev$vectors
 u <- as.matrix(abs(Re(U[, lmax]))/sum(abs(Re(U[, lmax]))))
 u <- u/(sum(u)) #scale u
 v <- as.matrix(abs(Re(V[, lmax])))
 v <- v/sum(u*v ) #scale v
 v <- t(ifelse(u*v <= 0, 0, v))
 return(list("lambda"=max(Re(ev$values)),"u"=u,"v"=v))
}
```

Bird example

For the bird example with a pre-reproductive census, we can calculate and plot the asymptotic properties as follows:

```
res.bird <- uvlambda(Abird.pre)
lambda.bird.pre <- res.bird$lambda
u.bird.pre <- res.bird$u
v.bird.bird <- res.bird$v

#Create data frame for plotting
birdframe.asym <- data.frame("AgeClass"=1:3,
                             "u"=u.bird.pre,
                             "v"=t(v.bird.bird))

#Long format for plotting:
birdframe.asym.long <- birdframe.asym %>%
  pivot_longer(c(u,v),
               names_to = "Vector",
               values_to = "Value")

#Ensure order of factor levels (for plotting)
birdframe.asym.long$AgeClass <- factor(birdframe.asym.long$AgeClass,
                                       levels=c(1,2,3))

ggplot(birdframe.asym.long) +
  geom_col(aes(x=AgeClass, y=Value,
               fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(Vector), ncol=2, scales="free")+
```

```
  scale_fill_manual(values=colors3)+ #Add colors manually
  labs( x="Age class", y="Value")+ #Axis labels
  theme(legend.position = "top" ) #Place legend on top
```

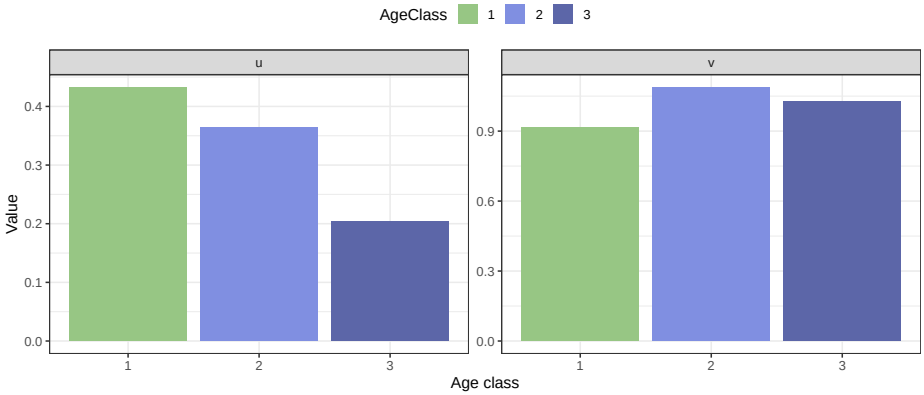


Figure 3.5: Stable age structure and reproductive values, calculated for the bird example with pre-reproductive census.

The calculated long-term growth rate is $\lambda = 1.0703$, corresponding well to the earlier result $\lambda = 1.0703$ calculated using projection.

Now we can also double check that we get the same value of λ for the pre-reproductive and post-reproductive bird models:

```
lambda.bird.pre <- uvlambda(Abird.pre)$lambda
lambda.bird.post <- uvlambda(Abird.post)$lambda
lambdas <- c(lambda.bird.pre, lambda.bird.post)
names(lambdas) <- c("Pre", "Post")
lambdas
```

```
##      Pre     Post
## 1.070261 1.070261
```

3.6.3 Life expectancy

From the matrix $\mathbf{U}$ we can calculate the socalled fundamental matrix, from which we can find the life expectancy of each age class:

$$\mathbf{N} = (\mathbf{I} - \mathbf{U})^{-1}, \tag{3.8}$$

where $\mathbf{I}$ is an identity matrix (a matrix with 1's on the main diagonal and 0's everywhere else) of the same dimension as $\mathbf{F}$ and $\mathbf{U}$.

The column sums of this matrix provides the life expectancy (expected remaining lifetime) in each age class. The life expectancy at birth is given by the first element of the vector, if we assume a post-reproductive census model.

To calculate the vector of age class-specific life expectancies in R we can use the following function:

```
LifeExpectancy <- function(MatU){
  k<-dim(MatU)[1]
  if(is.na(MatU[k,k])){
    MatU[k,k]<-0
  }
  uDim=dim(MatU)[1]
  N = solve(diag(uDim[1])-MatU)
  colSums(N)
}
```

Bird example

Using the post-reproductive model for the bird as an example, we get the following life expectancies:

```
#Define MatU for the post-reproductive model:
Ubird.post <- Abird.post
Ubird.post[1,] <- 0

#Calculate life expectancy vector
 LifeExpectancy.bird.post <- LifeExpectancy(MatU=Ubird.post)
```

For the bird example (post-reproductive census), the age-specific life expectancy is [1.38, 1.9, 1] years. Note that the life expectancy in age class 2 (actual age 1) is higher than in age class 1 (age 0). This is because many individuals die during their first year. The life expectancy in age class 3 (age 2) is exactly one year, because all individuals die at age 3 (after reproducing).

3.6.4 Lifetime reproduction

Using the submatrices **U** (survival matrix) and **F** (fertility matrix), we can calculate the net reproductive rate R_0 (mean lifetime reproduction), as well as the expected remaining reproduction of each class.

These calculations depend on the 'generation to generation projection matrix'

$$\mathbf{R} = \mathbf{F}(\mathbf{I} - \mathbf{U})^{-1} = \mathbf{FN} \tag{3.9}$$

Note that the fundamental matrix (equation **??**eq:Fundamental)) is also included in this definition. The matrix **R** describes the generation to generation growth of each age class. The column sums of **R** represent the expected remaining lifetime reproduction of the age class (with age structure, only the first row elements will be non-zero - but in more general stage structured models with more than one type of offspring, the matrix becomes more complex). R_0 is the dominant eigenvalue of the matrix **R**, just like λ is the dominant eigenvalue of **A** (remember that R_0 is the population growth rate per generation).

In R, we can use the following function to calculate R_0 and the column sums of the fundamental matrix, given the survival matrix **U** and the fertility matrix **F**.

```
R0function  <-  function(MatU, MatF){
  k <- dim(MatU)[[1]]
  Rmat <- MatF%*%solve(diag(1, k, k)-MatU)
  Rvec <- apply(Rmat,2,sum)
  R0 <- uvlambda(MatA=Rmat)$lam
  return(list("R0"= R0,"Rvec"= Rvec))
}
```

Bird example

For the bird example with a pre-reproductive census, we find these matrices and R_0 as follows:

```
Ubird.pre <- Fbird.pre <- Abird.pre
Ubird.pre[1,] <- 0
Fbird.pre[2,] <- 0
Fbird.pre[3,] <- 0

R0.Bird <- R0function(MatU=Ubird.pre, MatF=Fbird.pre)
```

For the bird example, R_0 is 1.19 and the column sums (remaining lifetime reproduction of each class) are 1.19, 1.32, 1.2.

3.6.5 Generation time

We can calculate another measure of generation time from the Leslie matrix and the fertility matrix, using the formula

$$G = \frac{\lambda}{\mathbf{vFu}}. \tag{3.10}$$

This measure is slightly different from the cohort generation time defined for the life table (equation (2.3)), and it represents the average age of mothers assuming stable structure. It relies on the assumption of a stable structure, notice how both **u** and **v** are part of the formula.

In R, we can use the following function to calculate this measure based on the projection matrix **A** and the fertility matrix **F**:

```
GenTime  <-  function(MatA, MatF){
  res <- uvlambda(MatA=MatA)
  lam <- res$lam
  u <- res$u
  v <- res$v
  lam/(v%*%MatF%*%u)
}
```

Bird example

```
GenTimeBird <- GenTime(MatA=Abird.pre, MatF=Fbird.pre)
```

For the bird example (pre-reproductive census), the generation time is 2.53 years.

3.7 Sensitivity and elasticity of λ

We will now turn to another important concept in life history theory: the **sensitivities** (and **elasticities**) of λ with respect to elements of the Leslie matrix. The sensitivities define the **selection gradients** on age-specific survival and fertility, reflecting the strength of selection on different parts of the life cycle. They also play a key role in conservation biology and in other calculations such as life table response experiments (section 3.9), and population growth rate in stochastic environments (chapter 5).

3.7.1 Sensitivity to matrix elements

The sensitivity of λ to a projection matrix element A_{ij} (here i refers to row and j refers to column of **A**) is given by the partial derivative

$$\frac{\partial \lambda}{\partial A_{ij}} = v_i u_j, \tag{3.11}$$

where the repreoductive values are scaled so that $\mathbf{vu} = 1$.

We can calculate the **sensitivity matrix** as $\mathbf{S} = (\mathbf{uv})^{\mathbf{T}}$ (remember that **u** is a column vector and **v** is a row vector, and "T" denotes the transpose). Each element of the sensitivity matrix corresponds to the sensitivity $S_{ij})v_i u_j$. The following R code does this calculation for a given Leslie matrix **A**, by first calculating the stable age structure and reproductive values (using our previously defined function ´uvlambda()´), and then calculating the sensitivities:

```
sensitivity.matrix <- function(MatA, zeroes=T){
  res <- uvlambda(MatA=MatA)
  sensmat <- t(res$u%*%res$v)
  if (zeroes==T){
    sensmat <- ifelse (MatA==0, 0, sensmat)
  }
 sensmat
}
```

With the default argument `zeroes=T`, all sensitivity matrix elements that correspond to elements of **A** that are zero, are also set to zero. If we set `zeroes=F`, the function will return a sensitivity value for each element of the Leslie matrix, including the zero elements representing for instance transitions from age 2 to age 5 in one year which are clearly not realistic (one can still calculate the sensitivity of λ for such transitions; the model does not 'care' about realism). Since we are usually only interested in the realistically possible transitions, it is common to set the unrealistic sensitivities to zero.

Bird example

Applying the code on the bird example with post reproductive census, we get the following sensitivity matrix:

```
sens.mat.post <- sensitivity.matrix(MatA=Abird.post)
round(sens.mat.post,3)
```

```
##       [,1]  [,2]  [,3]
## [1,] 0.000 0.074 0.062
## [2,] 2.116 0.000 0.000
## [3,] 0.000 0.249 0.000
```

We can plot the corresponding sensitivities to survival and fertility coefficients:

```
 #Extract fertility sensitivities:
fert.sens <- sens.mat.post[1,]

#Extract survival sensitivities:
surv.sens <- c(sens.mat.post[2,1],
               sens.mat.post[3,2],
               sens.mat.post[3,3])
```

```
#Create data frame for plotting
birdframe.sens <- data.frame("AgeClass"=1:3,
                             "FertilitySensitivity"=fert.sens,
                             "SurvivalSensitivity"=surv.sens)

#Long format for plotting:
birdframe.sens.long <- birdframe.sens %>%
  pivot_longer(c(-"AgeClass"),
               names_to = "SensitivityTo",
               values_to = "Value")

birdframe.sens.long$AgeClass <- factor(birdframe.sens.long$AgeClass,
                                       levels=c(1,2,3))
#Plot

ggplot(birdframe.sens.long) +
  geom_col(aes(x=AgeClass, y=Value,
               fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(SensitivityTo),ncol=2 )+
  scale_fill_manual(values=colors3)+ #Add colors manually
  labs( x="Age class", y="Sensitivity")+ #Axis labels
  theme(legend.position = "top" ) #Place legend on top
```

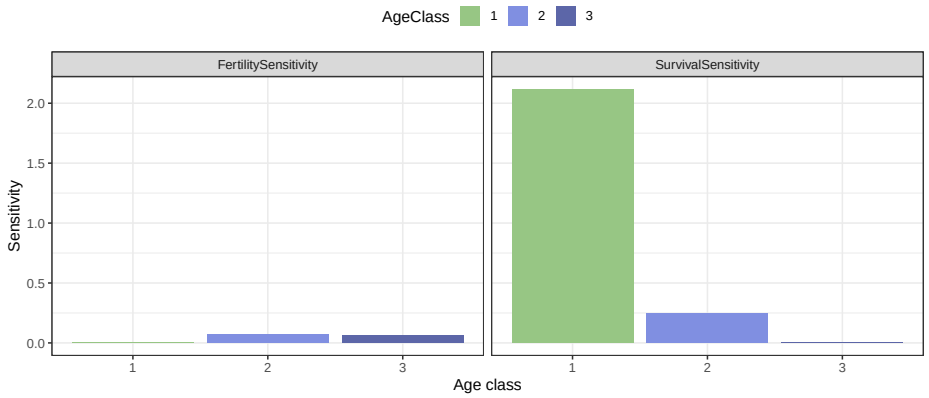


Figure 3.6: Sensitivities of lambda to fertility and survival coefficients, for the bird example with post-reproductive census.

For this example, we see that the highest sensitivities are found for the survival probabilities, and in particular the first one corresponding to survival from age 0 to 1 (since we used a post-reproductive census in this example).

3.7.2 Sensitivity to underlying parameters

We can also calculate the sensitivity of λ to underlying (lower-level) parameters in the projection matrix, by applying the chain rule of derivation. If a parameter occurs in multiple entries of the matrix, we need to sum up all the corresponding sensitivities. In general, the sensitivity of λ to a lower level parameter β can be defined as

$$\frac{\partial \lambda}{\partial \beta} = \sum_{i,j} \frac{\partial \lambda}{\partial A_{ij}} \frac{\partial A_{ij}}{\partial \beta}. \tag{3.12}$$

This kind of analysis is useful in models where the matrix elements are functions of some environmental variable, like temperature. Then we can calculate the sensitivity of λ to temperature through each matrix element (or sum up to age class), to understand more about how temperature affects fitness.

Bird example

For instance, consider the projection matrix of the bird example with a post-reproductive census (equation (3.7)):

$$\mathbf{A} = \begin{bmatrix} f_1 & f_2 & f_3 \\ s_1 & 0 & 0 \\ 0 & s_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & p_1 m_2 & p_2 m_3 \\ p_0 & 0 & 0 \\ 0 & p_1 & 0 \end{bmatrix}. \tag{3.13}$$

If we want to calculate the sensitivity of λ to the parameter p_1 (which enters element A_{12} and A_{32} in the projection matrix), we get the following expression:

$$\begin{aligned} \frac{\partial \lambda}{\partial p_1} &= \sum_{i,j} \frac{\partial \lambda}{\partial A_{ij}} \frac{\partial A_{ij}}{\partial p_1} && (3.14) \\ &= v_1 u_2 \frac{\partial A_{12}}{\partial p_1} + v_3 u_2 \frac{\partial A_{32}}{\partial p_1} && (3.15) \\ &= v_1 u_2 m_2 + v_3 u_2. && (3.16) \end{aligned}$$

We can use the following code to calculate and plot the sensitivites of λ to the survival probabilities and fecundities against actual age, assuming the post-reproductive census:

```
res <- uvlambda(Abird.post)
v <- res$v
u <- res$u

#Fecundity sensitivities m0-m1-m2-m3:
m.sens <- c(0,
            0,
            v[1]*u[2]*px[2],
            v[1]*u[3]*px[3])

#Survival sensitivities p0-p1-p2-p3
p.sens <- c(v[2]*u[1],
            v[1]*u[2]*mx[3]+v[3]*u[2],
            v[1]*u[3]*mx[4],
            0)

sens.table.bird <- data.frame("Age"=0:3,
                              "mSensitivity"=m.sens,
                              "pSensitivity"=p.sens)


#Long format for plotting:
sens.table.bird.long <- sens.table.bird %>%
  pivot_longer(c(-"Age"),
               names_to = "SensitivityTo",
               values_to = "Value")

sens.table.bird.long$Age<- factor(sens.table.bird.long$Age,
                                  levels=c(0,1,2,3))

#Plot

ggplot(sens.table.bird.long) +
  geom_col(aes(x=Age, y=Value, fill=Age ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(SensitivityTo),ncol=2 )+
  scale_fill_manual(values=colors4)+ #Add colors manually
  labs( x="Age (years)", y="Sensitivity")+ #Axis labels
  theme(legend.position = "top" ) #Place legend on top
```

3.7.3 Elasticities

A closely related concept to sensitivity is the **elasticity**. The elasticity of λ to a projection matrix element A_{ij} is defined as the proportional change in λ to a

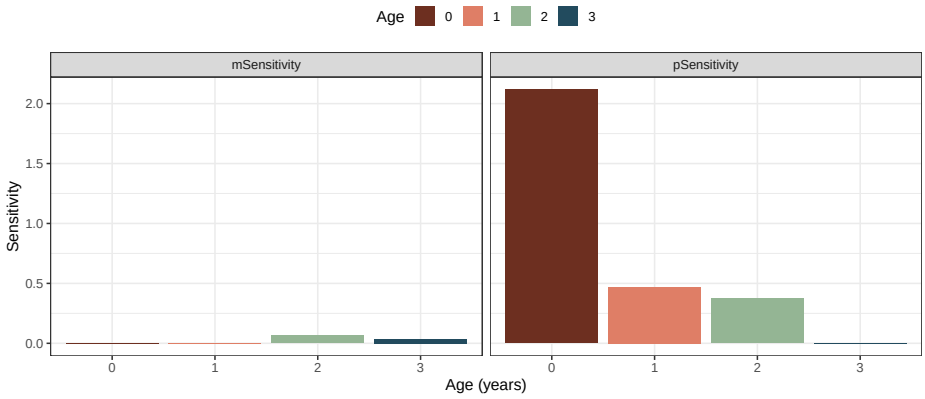


Figure 3.7: Sensitivities of lambda to underlying parameters of age-specific fecundity and survival probability, for the bird example with post-reproductive census.

proportional change in the matrix element, and is given by

$$\frac{\partial \ln \lambda}{\partial \ln A_{ij}} = \frac{A_{ij}}{\lambda}\frac{\partial \lambda}{\partial A_{ij}} = \frac{A_{ij}}{\lambda} v_i u_j = \frac{A_{ij}}{\lambda} S_{ij} = . \tag{3.17}$$

The function below returns the elasticity matrix for a given projection matrix **A**.

```
elasticity.matrix <- function(MatA, zeroes=T){
  res <- uvlambda(MatA=MatA)
  sensmat <- t(res$u%*%res$v)
  if (zeroes==T){
    sensmat <- ifelse (MatA==0, 0, sensmat)
  }
 sensmat*MatA/res$lam
}
```

The elasticity to a lower-level parameter β can be found in a similar way, as

$$\frac{\partial \ln \lambda}{\partial \ln \beta} = \frac{\beta}{\lambda}\frac{\partial \lambda}{\partial \beta}. \tag{3.18}$$

Bird example

Continuing with the bird example with a post-reproductive census, the elasticities to age-specific fecundity and survival probability can be found as follows, using the sensitivities found in the previous section:

```
#Fecundity elasticities m0-m1-m2-m3:
m.elas <- mx/lambda.bird.post*m.sens

#Survival elasticities p0-p1-p2-p3
p.elas <- px/lambda.bird.post*p.sens

elas.table.bird <- data.frame("Age"=0:3,
                              "mElasticity"=m.elas,
                              "pElasticity"=p.elas)

#Long format for plotting:
elas.table.bird.long <- elas.table.bird %>%
  pivot_longer(c(-"Age"),
               names_to = "ElasticityTo",
               values_to = "Value")

elas.table.bird.long$Age<- factor(elas.table.bird.long$Age,
                                  levels=0:3)
#Plot
ggplot(elas.table.bird.long) +
  geom_col(aes(x=Age, y=Value, fill=Age ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(ElasticityTo),ncol=2 )+
  scale_fill_manual(values=colors4)+
  labs( x="Age (years)", y="Elasticity")+
  theme(legend.position = "top" )
```

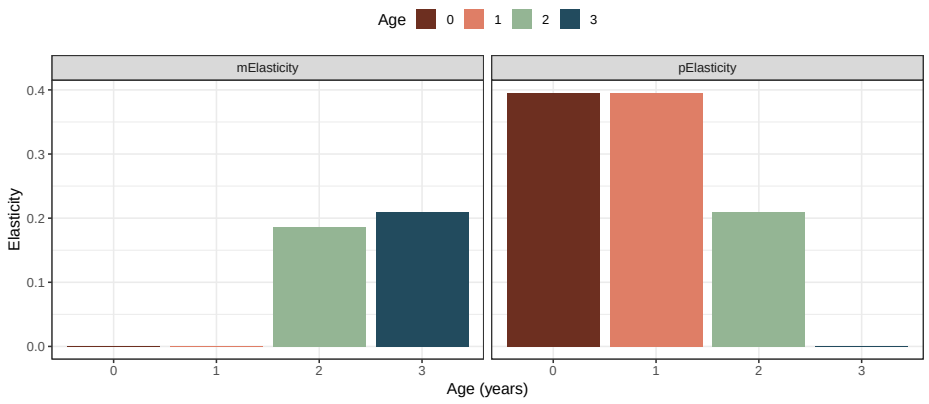


Figure 3.8: Elasticities of lambda to underlying parameters of age-specific fecundity and survival probability, for the bird example with post-reproductive census.

Note that the elasticities of fecundities m_x and survival probabilities p_x are

more similar than the sensitivites. This is because the elasticities represent proportional changes in λ while sensitivities represent absolute changes, and fecundity and survival are variables with different scales. We should always keep this in mind when interpreting sensitivities (and elasticities), as an absolute change in fecundity is not the same as the same absolute change in a survival probability. Although elasticities measure proportional effects, they are not better than sensitivities in this sense (which is sometimes claimed), and still need to be interpreted carefully.

3.8 Senescence

The sensitivities of λ with respect to age-specific survival and fecundities are important because (among other things) they measure selection pressures. More generally, a **selection gradient** of a trait (survival probability and fertility are also traits) is defined as the slope of fitness as a function of the trait. For a trait with value θ, the selection gradient is the partial derivative:

$$\text{Gradient} = \frac{\partial \text{Fitness}}{\partial \theta}. \tag{3.19}$$

If the trait is a vector, such as the vector of age-specific survival probabilities p_i, the gradient is also a vector.

Based on the Euler-Lotka equation, Hamilton [1966] derived the selection gradient of $r = \ln \lambda$ on (log) age-specific survival probability p_x:

$$\frac{\partial r}{\partial \ln p_x} = \frac{\sum_{y=x}^{k} e^{-ry} l_y m_y}{T_C}, \tag{3.20}$$

where T_C is the cohort generation time as defined in chapter 2. The selection gradient on age-specific fecundity m_x is given by

$$\frac{\partial r}{\partial m_x} = \frac{e^{-rx} l_x}{T_C}. \tag{3.21}$$

The selection gradient on survival will decline with age x, after the age at first reproduction. For fecundity, the selection gradient will usually decrease with age, except in strongly declining populations: If r is sufficiently negative relative to survival probability so that the stable age distribution increases with age, then the selection gradient can increase. This situation is unlikely to be common or persist for long time periods.

Because the selection gradients (nearly) always decline with age, Hamilton [1966] concluded that senescence is inevitable for any conceivable organism. With his selection gradients, Hamilton formalized the arguments previously given regarding mutation accumulation hypothesis [Medawar, 1952], and the antagonistic pleiotropy hypothesis [Williams, 1957]. Any change in the environment that causes the selection gradients to fall off more rapidly with age should select for increased senescence, while changes that make the slope less steep should select for reduced senescence.

Bird example

We can plot these selection gradients for the bird example (note that we now use the life table from chapter 2):

```
res <- uvlambda(Abird.post)
r <- log(res$lam)

#Survival gradient:
dr.dp <- rev(cumsum(rev(exp(-r*x)*lx*mx)))/TC_Bird

#Fecundity gradient:
dr.dm <-  exp(-r*x)*lx/TC_Bird

bird.table <- data.frame(x, dr.dp, dr.dm)

#long format for plotting:
bird.table.long <- bird.table %>% pivot_longer(c( dr.dp, dr.dm),
               names_to = "Sensitivity", values_to = "Value")

bird.table.long$Age <- factor(bird.table.long$x, levels=0:3)

ggplot(bird.table.long) +
  geom_col(aes(x=Age, y=Value , fill=Age) , lwd=1.2)+
  facet_wrap(vars(Sensitivity))+
  scale_fill_manual(values=colors4)+
  theme_bw() +
  labs( x="Age (year)", y="Sensitivity of r")+
   theme(legend.position = "none" )
```

We see that also for the bird example the selection gradients decline with age, $\frac{\partial r}{\partial p_x}$ after the age of first reproduction (here 2 years) and $\frac{\partial r}{\partial m_x}$ for all ages.

The effect of extrinsic mortality on senescence

Williams [1957] suggested that increased extrinsic mortality could be such a fac-

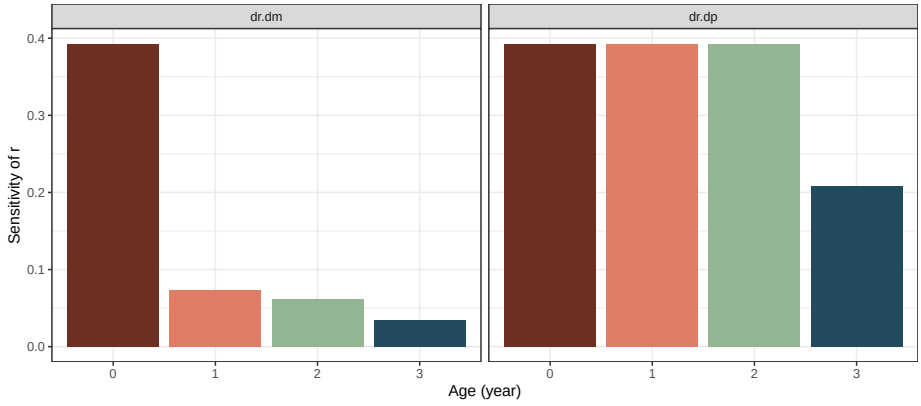


Figure 3.9: Selection gradients on fecundity and survival probability in the bird example, based on the life table from chapter 2.

tor leading to steeper gradients, and that simply living in a more risky environment (e.g. with many predators) should select for more rapid senescence. This may seem intuitive, however other researchers have shown that if the extrinsic mortality is age-independent, it has no effect on the slope of the selection gradient and thus no effect on senescence [Hamilton, 1966, Caswell, 2007, Wensink et al., 2017]. The reason is that extra mortality will affect not only the the survivorship l_x, but also parameters that depend on it such as r and the stable age structure. It turns out that when the added mortality is age-independent, effects cancel each other out in the numerator and denominator of Hamilton's selection gradient, so that there is no effect on senescence. If the extrinsic mortality is age-dependent, however (for instance if only younger individuals are exposed to increased predation), the selection gradients and senescence will also be affected. In general (and all else being equal), increased mortality in early life will then select for reduced senescence, while increased mortality in late life selects for increased senescence.

An associated common misconception is that senescence occurs 'because few individuals survive to old age', and that old-age individuals are therefore evolutionarily unimportant. This argument ignores the role of reproduction: Patterns of senescence cannot be described by declines in survivorship with age alone (the l_x curve; see chapter 2), because life histories are shaped by both survival and reproduction. Think of the life history of a large tree species: Only few individuals survive from seedling to old age, yet senescence is hardly present in many of these species. Even in the hypothetical situation of a species with no mortality at all (a flat l_x-curve), the selection gradients would still decline with age, because of reproduction leading to ever-increasing numbers of newborn individuals compared to older ages.

3.9 Life table response experiments (LTRE)

Sometimes we want to compare larger differences in λ between two or more different conditions - for instance for populations living in different environments, or populations that have experienced different management treatments. The purpose of a life table response experiment (LTRE) is to understand how different projection matrix elements (or underlying parameters) have contributed to the difference in λ. Caswell [2001], who introduced this method, describes it in more detail. LTRE analysis is inspired by experimental approaches and analysis of variance (ANOVA), but does not require any actual experiments to take place. The 'experiment' can also be for instance an observed change in the projection matrix after a sudden environmental change, or a management decision.

3.9.1 General description of LTRE approach

To do an LTRE analysis to compare λ in two environments, we first define a **reference matrix** $\mathbf{A}^{(r)}$, defining the reference life history with growth rate $\lambda^{(r)}$. Now assume that we want to compare this to another matrix $\mathbf{A}^{(m)}$ for the same species, where the vital rates are different e.g. because of some environmental impact. This matrix is called a **treatment matrix** and has a corresponding $\lambda^{(m)}$. We want to understand how much of the difference between $\lambda^{(m)} - \lambda^{(r)}$ is coming from each vital rate. To do this, we first define two useful matrices [Caswell, 2001]:

- $\mathbf{A}^* = \frac{1}{2}\left(\mathbf{A}^{(m)} + \mathbf{A}^{(r)}\right)$ defines the 'halfway matrix', i.e. a matrix where each element is the average of the corresponding elements in $\mathbf{A}^{(m)}$ and $\mathbf{A}^{(r)}$.
- $\mathbf{D} = \mathbf{A}^{(m)} - \mathbf{A}^{(r)}$ is the 'difference matrix', where each element corresponds to the difference between the corresponding elements in $\mathbf{A}^{(m)}$ and $\mathbf{A}^{(r)}$.

Now we can write $\lambda^{(m)}$ as a function of $\lambda^{(r)}$ (the reference value), plus a sum of difference contributions from each matrix element, weighted by the senstivity of λ to that element [Caswell, 2001]:

$$\lambda^{(m)} \approx \lambda^{(r)} + \sum_{i,j} d_{ij} \left.\frac{\partial \lambda}{\partial a_{ij}}\right|_{\mathbf{A}^*}, \tag{3.22}$$

where the sensitivities are calculated for the matrix $\mathbf{A}^*$. A common output of such an analysis is to plot these contributions side by side. Doing a life table response analysis for λ involves the following steps:

1) Define all the relevant projection matrices for each treatment / environment and define a reference matrix, and calculate λ for each.

2) Calculate $\mathbf{A}^*$ and $\mathbf{D}$,

3) Calculate the sensitivity matrix corresponding to the matrix $\mathbf{A}^*$,

4) Calculate the contributions from each vital rate to the difference in λ according to the sum defined above.

Note that LTRE analyses can also be done for more than one comparison environment, using the same approach as described here.

3.9.2 LTRE and sensitivity analyses

An LTRE analysis is *retrospective*, aiming to explain observed differences in λ, while the sensitivity analysis is *prospective*, aiming to explain potential future responses of λ to perturbations of the matrix elements [Caswell, 2001]. The two can give very different answers in terms of which matrix element is the 'most important' (largest sensitivity or largest LTRE contribution).

Bird example

We can demonstrate the LTRE analysis using the bird example. Here we will again assume a pre-reproductive census (you can do the same analysis for the post-reproductive model). The matrix defined in equation (3.6) is now the reference matrix:

$$\mathbf{A}^{(r)} = \begin{bmatrix} 0 & 0.6 & 1.2 \\ 0.9 & 0 & 0 \\ 0 & 0.6 & 0 \end{bmatrix}.$$

Remember that with this projection matrix we get a value of $\lambda^{(r)} \approx 1.07$.

Now assume that the bird was affected by increased predation, leading to a different projection matrix

$$\mathbf{A}^{(m)} = \begin{bmatrix} 0 & 0.4 & 1 \\ 0.8 & 0 & 0 \\ 0 & 0.4 & 0 \end{bmatrix}.$$

For this treatment matrix we can calculate the following value of $\lambda^{(m)}$ and its difference to the reference:

```
#Reference matrix:
AmatBirdR <-  Abird.pre

#Define treatment matrix:
AmatBirdM <- Create.Amat(Svec=c(.8,.5,.0), Fvec=c(0,.5,1 ))

#Lambda in reference environment:
lambdaR<- uvlambda(AmatBirdR)$lam

#Lambda in treatment environment
lambdaM <- uvlambda(AmatBirdM)$lam

#Difference in lambda:
lamDiff <- lambdaM-lambdaR
```

In the environment with increased predation, we get $\lambda^{(m)} \approx 0.915$. This leads to a quite large difference of $\lambda^{(m)} - \lambda^{(r)} \approx$ -0.155. Now we are interested in which matrix element contributes the most to this difference in λ.

First we calculate the halway matrix and difference matrix:

```
HalfwaymatBird <- (AmatBirdM+AmatBirdR)/2
DiffmatBird <- AmatBirdM-AmatBirdR

DiffmatBird
```

```
##      [,1] [,2] [,3]
## [1,]  0.0 -0.1 -0.2
## [2,] -0.1  0.0  0.0
## [3,]  0.0 -0.1  0.0
```

The largest difference is found for the fertility of age class 3. But this is not necessarily the element contributing the most to the difference in λ.

To calculate the contributions to λ we first need the sensitivity matrix calcalated for the halfway matrix $\mathbf{A}^*$. Then we can apply equation (3.22) and plot the components of this sum:

```
SensHalfwaymatBird <- sensitivity.matrix(HalfwaymatBird)

#LTRE contributions:
LTREMat <- DiffmatBird*SensHalfwaymatBird

FertilityContributions <- LTREMat[1,]
SurvivalContributions <- c(LTREMat[2,1],
                           LTREMat[3,2],
                           LTREMat[3,3])
```

```
LTRE.table <- data.frame(AgeClass=factor(1:3),
                         FertilityContributions,
                         SurvivalContributions)

#long format for plotting:
LTRE.table.long <- LTRE.table %>%
  pivot_longer(c( FertilityContributions,
                  SurvivalContributions),
               names_to = "Component",
               values_to = "Value")

ggplot(LTRE.table.long) +
  geom_col(aes(x=AgeClass, y=Value, fill=AgeClass), lwd=1.2)+
  facet_wrap(vars(Component))+
  scale_fill_manual(values=colors3)+
  theme_bw() +
  labs( x="Age class", y="Contribution to lambda difference")+
    theme(legend.position = "none" )
```

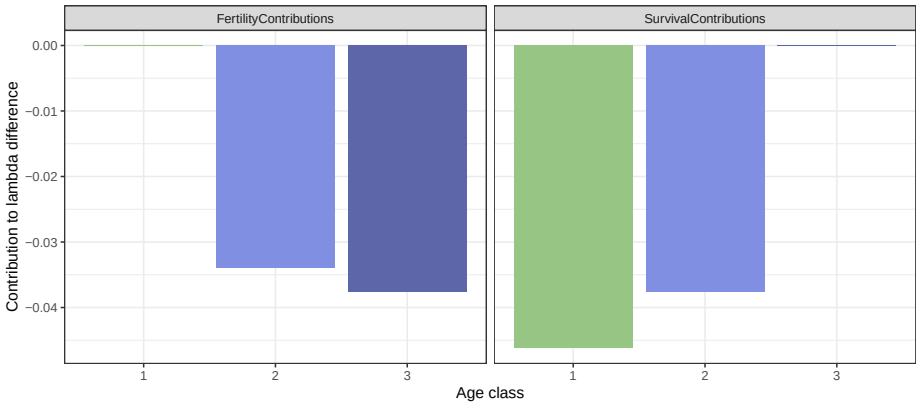


Figure 3.10: LTRE components for the bird example, corresponding to fertility and survival coefficients in the prereproductive census model, compared to an alternative model with increased predation.

Here we see that the element contribution most to the difference $\lambda^{(m)} - \lambda^{(r)}$ is the survival probability of age class 1, followed by fertility of age class 3, survival of age class 2, and fertility of age class 2.

We can check the approximation by summing up the LTRE contributions:

```
sumDiff <- sum(LTREMat)
Vek <- c(lamDiff, sumDiff)
names(Vek) <- c("Exact", "LTREsum")
```

```
Vek
```

```
##      Exact    LTREsum
## -0.1552903 -0.1552733
```

These values are quite similar, indicating that the LTRE approximation is ok.

In this example predation had a negative effect on projection matrix elements, but it is often the case that the effect of a perturbation is negative on some elements, and positive on others. The total difference in λ can then be quite small despite large effects on each matrix element, and an LTRE analysis is important to reveal the opposing effects.

3.10 Exercises

For suggested solutions, see appendix A.

Exercise 3.1

Draw the post-reproductive life cycle graph for the bird example (projection matrix shown in equation (3.7)).

Exercise 3.2

Start with the complete life table you calculated in Exercise 2.2.

1. Draw a life cycle graph for this population (on paper or using a drawing program), assuming a **pre-reproductive census**.
2. Calculate the vector of fertility coefficients f_i and survival probabilities s_i, assuming a pre-reproductive census.
3. Define the Leslie matrix **A** in R for this population (still assuming pre-reproductive census), as well as its two components **U** and **F**.
4. Use the Leslie matrix to project the growth of each age class over 30 time steps, and plot the result.
5. Calculate the long-term growth rate λ, the net reproductive rate R_0 and the generation time **G** using the three matrices you defined in task 3. Do you get the same values as in exercise 2.2? If not, can you explain the difference?
6. Calculate and plot the stable age structure **u** and reproductive values **v** for the population. Explain the biological meaning of the results.

7. Assume that you could increase reproduction by multiplying the entire vector of fecundities m_x by some constant factor c. How large should this factor be to achieve a long-term growth rate λ above 1? Update the three matrices **A**, **U** and **F** using the new fecundity values (give them new names in R) and do the same analysis as above for the new model. What happened to R_0 and G? How did the resulting stable structure and reproductive values change?

8. This time increase the annual survival probability vector p_x by some factor, instead of m_x, to get a $\lambda > 1$. Is the resulting survival probability vector biologically realistic? Why/why not?

9. Do the exercise of point 1-8 again, but this time change from a pre-reproductive to post-reproductive census model. Which results have now changed, and which have not?

Chapter 4

Stage structured models

The life table and associated parameters defined in the previous chapters apply only to age structured populations. However, the methods you learned for matrix models can be used with any projection matrix **A** (provided it fulfills certain mathematical assumptions). The only difference is that the results refer to stages instead of age class. This chapter will present a general stage structured matrix model and show how to build the projection matrix from four main components describing transitions, survival probability and fertility. We will then go through some main sources of stage structure and consider examples.

4.1 Learning goals

- Understand the key components of a stage structured matrix model.
- Get the vectors of survival probability and fertility coefficients from the matrices **U** and **F** .
- Discuss some main sources of stage structure.
- Draw a life cycle graph for any stage structured model.
- Apply methods from chapter 3 to stage structured matrix models
- Define a simple integral projection model for size structured populations.

4.2 General stage structured matrix model

We will now extend the age structured model to the general case of stage structured models, which can be adapted to describe many different kinds of stage

structure. As shown by Vindenes et al. [2020], the general projection matrix $\mathbf{A}$ can be decomposed into four main components:

$$\mathbf{A} = \mathbf{U} + \mathbf{F} = \mathbf{G}\text{Diag}(\mathbf{s}) + \mathbf{Q}\text{Diag}(\mathbf{f}), \tag{4.1}$$

where $\mathbf{G}$ is a transition matrix containing the transition probabilities g_{ij} from stage j to stage i per time step (column sums of this matrix should all be 1), $\mathbf{s}$ is the vector of survival probabilities (survival from current to next time step) in each stage, $\mathbf{Q}$ is a matrix containing the probabilities q_{ij} that an offspring produced in stage j is assigned to stage i at birth (columns should sum to 1), and $\mathbf{f}$ is the vector of fertilities per stage, i.e. the mean number of offspring contributed by an individual to next time step's population.

For a population with 3 stages, the elements of each matrix in equation (4.1) are given by

$$\begin{aligned}
&\begin{bmatrix} g_{11}s_1 + q_{11}f_1 & g_{12}s_2 + q_{12}f_2 & g_{13}s_3 + q_{13}f_3 \\ g_{21}s_1 + q_{21}f_1 & g_{22}s_2 + q_{22}f_2 & g_{23}s_3 + q_{23}f_3 \\ g_{31}s_1 + q_{31}f_1 & g_{32}s_2 + q_{32}f_2 & g_{33}s_3 + q_{33}f_3 \end{bmatrix} \\
&= \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{33} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} + \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{33} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} f_1 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & f_3 \end{bmatrix}.
\end{aligned}$$

The stage-specific survival and fertility coefficients are the same as before: The survival coefficient s_j represent the probability that an individual in stage j will survive until the next time step. The fertility coefficient f_j represent the average number of offspring contributed by an individual in stage j to the population in the next time step. Census time still plays a role in the definition of the fertility coefficient from underlying parameters.

The main difference from age structured models is the explicit specification of the transition matrices $\mathbf{G}$ and $\mathbf{Q}$. The matrix $\mathbf{G}$ specifies the transition probabilities between stages in the model, while $\mathbf{Q}$ specifies the probabilities for assigning offspring among the stages.

Note that for the methods defined in chapter 3 to work, the transition matrices together need to be defined in such a way that all stages are 'connected'. For any given stage, it must be possible for an individual or its descendents to reach any other stage in the model within some (non-infinite) number of time steps. Otherwise, the model could lead to disconnected sub-populations and the methods for calculating λ, stable structure etc would no longer work. We will not define the mathematical criteria for these requirements here (see e.g. Caswell [2001] for details), but note that with stage-structured models we have to be more careful with specifying the stages and transition probabilities. For most biologically realistic and well defined models this is not a problem.

With this general definition of the projection matrix we can obtain special cases by adjusting the transition matrices **G** and **Q**. For instance, for an age structured model with 3 stages these matrices are given by

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix},$$

and

$$\mathbf{Q} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Note that the transition matrix **G** has the value 1 in the lower right corner, meaning that if the survival probability is not zero in the final age class individuals will be allowed to accumulate in this class. For a 'true' age-structured model the survival probability should be zero in the final class. Since all offspring are born into the first age class, the matrix **Q** contains 1's on all elements of the first row (even if the fertility coefficient may be 0) and 0 elsewhere.

4.3 Decomposing U and F

It is often useful to decompose the matrices **U** and **F** based on equation (4.1) to obtain the transition matrices **G** and **Q** and the vectors of survival probability **s** and fertility **f**.

Note that in general we need to do this decompososition from **U** and **F**, as it is not possible to decompose elements of **A** directly (without being given more information on the life history).

To decompose **U** and **F** we take advantage of the fact that the column sums of the transition matrices have to sum up to one (individuals have to end up in one of the stages, if they survive). This means that the column sums of **U** are the stage-specific survival coefficients, while the column sums of **F** are the stage-specific fertilities.

The following R function returns the survival vector **s** and the transition matrix **Q** based on the matrix **U**:

```
DecomposeU <- function(MatU){
  k<-dim(MatU)[1]
  if(is.na(MatU[k,k])){
    MatU[k,k]<-0
  }
```

```
  Svec <- apply(MatU,2,sum)
  MatG <- t(t(MatU)/Svec)
  for(i in 1:k){
    if(Svec[i]==0){
      MatG[,i] <- 0
      MatG[i,i] <- 1
    }
  }
 list("Gmat"=MatG,"Survival"=Svec)
}
```

Similarly, this following R function returns the survival vector **s** and the transition matrix **Q** based on the matrix **U**

```
DecomposeF <- function(MatF){
    k <- dim(MatF)[1]
    Fvec <- apply(MatF,2,sum)
    MatQ <- matrix(0,k,k)
    for(i in 1:k){
        if(Fvec[i]==0){
            MatQ[1,i]<-1

        }
        else{
            MatQ[,i]<-MatF[,i]/Fvec[i]
        }
    }
 list("Qmat"=MatQ,"Fertility"=Fvec)
}
```

4.4 Examples of stage structured models

This section includes some descriptions and examples for a few common types of stage structured models. This is by no means a comprehensive list of all possible kinds of stage structure - which is almost endless.

4.4.1 Lefkovitch models

Lefkovitch [1965] introduced an important extension of age structured models for life histories of discrete stages where individuals can either remain in the same stage or transition to the next stage with a certain probability. Distinct life cycle stages are typical for many invertebrates, and the Lefkovitch model can

also be used for discrete stages representing size, where individuals can either grow (to the next size stage) or not.

With this kind of structure, there will be two transition probabilities for each stage. For instance, if the the probability of developing from stage j to $j+1$ is denoted g_j, then the probability of remaining is $(1-g_j)$. Offspring transitions are usually the same as in age structured models, i.e. the first stage is the offspring stage and the matrix $\mathbf{Q}$ will have 1's on the first row and 0's elsewhere. The projection matrix of such a model with 3 stages takes the form

$$\mathbf{A} = \begin{bmatrix} (1-g_1)s_1 + f_1 & f_2 & f_3 \\ g_1 s_1 & (1-g_2)s_2 & 0 \\ 0 & g_2 s_2 & s_3 \end{bmatrix}. \tag{4.2}$$

This kind of projection matrix is also known as a **Lefkovitch matrix**. Here, surviving individuals accumulate in the final stage. Note also that the first element A_{11} of the transition matrix potentially depends on both survival and reproduction. Although in most life histories individuals do not reproduce during their first year, this can happen in some cases. For these models we need to be careful when decomposing the projection matrix so that fertility coefficients are not inadvertently interpreted as survival, and vice versa.

Example: Wild boar

Figure 4.1: Wild boar (*Sus scrofa*). Photo: Dave Pape

As an example of a study using a Lefkovitch matrix, we can consider the model of Bieber and Ruf [2005] for female wild boar. Their model has annual time steps and three stages: Juveniles (age 0-1), yearlings (age 1-2) and adults (age 2+). Thus, the first to stages represent age classes lasting 1 year each, while the final stage contain all individuals of age 2 and above. Wild boar start reproducing

before their first year, so in this case even the juveniles can contribute offspring to the next year's population (figure 4.2).

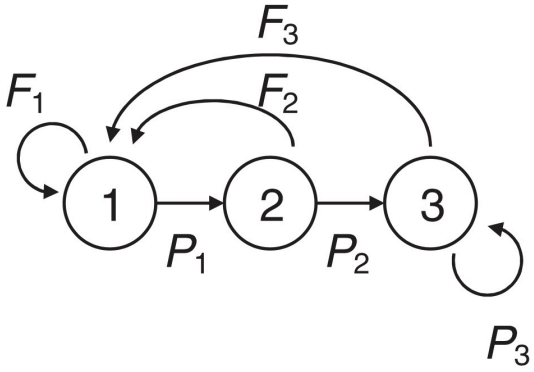


Figure 4.2: Life cycle graph for the wild boar (*Sus scrofa*), fig 1 from @Bieber1.

The projection matrix for this wild boar model is given by

$$\mathbf{A} = \begin{bmatrix} 0.26 & 0.94 & 1.93 \\ 0.33 & 0 & 0 \\ 0 & 0.4 & 0.66 \end{bmatrix}.$$

This matrix is quite similar to an age structured model (Leslie matrix), but with the exception that adults do not necessarily die after reaching age 2, but can continue to live so that the final stage contains all individuals of age 2 and more (2+).

4.4.2 Age by stage models

Individuals in a population will always differ in more than one way, and sometimes (depending on the study question as well as the species) we have to account for more than one state variable in order to adequately describe the life history. For instance, we may need to combine age and life cycle stage, if the age differences within each stage are large. Even if age does not have a large effect on survival and fecundity compared to stage, we often need to describe how model outputs (e.g. sensitivities) change with age, for instance to compare with other age-structured models. Models that combine age and stage are able to handle a wide range of possible structures - and can be seen as a special case of the general stage structured model where each class is a combination of age and stage.

For instance, there are k age classes, and m stages, the new age by stage model will have $\phi = k * m$ classes in total. From these classes we can define a new

projection matrix of dimension $\phi \times \phi$, which can be analysed using the same methods as before to find λ and other parameters. For a broad overview of this kind of matrix model, see Caswell et al. [2018].

Example: Black-headed gulls

Figure 4.3: Black-headed gull (*Chroicocephalus ridibundus*). Photo: Hans Hillewaert (https://creativecommons.org/licenses/by-sa/4.0/)

Lebreton [1996] used an age-by-stage model to study a french colony of black-headed gulls (*Chroicocephalus ridibundus*) living in two types of habitat of 'good' and 'poor' quality. In this species dispersal among habitats happens mainly before first reproduction, but young breeders may also disperse. Lebreton [1996] considered different scenarios for dispersal, here we will consider the scenario where only offspring disperse. This model was also used as an example by Hunter and Caswell [2005] demonstrating a useful approach ('vec-permutation') to calculate sensitivity in age-by-stage models, and by Vindenes et al. [2020] in their introduction to matrix models.

The gulls have five main age classes corresponding to age 1, 2, 3, 4, and 5+ (all individuals of age 5 and older). With two habitats and five age classes we get a 10×10 projection matrix, where each stage corresponds to a combination of habitat and age class. There are two ways to structure such a matrix [Lebreton, 1996]: Either include age within habitat (so that the first five stages represent habitat 1 and the last five habitat 2), or habit within age class (the first two stages represent habitat 1 and 2 in age class 1, etc).

Using a structure of age within habitat, we get the following projection matrix [Vindenes et al., 2020]:

$$\mathbf{A} = \left[\begin{array}{ccccc|ccccc} f_1 & f_2 & f_3 & f_4 & f_5 & 0 & 0 & 0 & 0 & 0 \\ (1-p)s_1 & 0 & 0 & 0 & 0 & qs_6 & 0 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & s_4 & s_5 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & f_6 & f_7 & f_8 & f_9 & f_{10} \\ ps_1 & 0 & 0 & 0 & 0 & (1-q)s_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_9 & s_{10} \end{array}\right]. \quad (4.3)$$

The constants p and q describe dispersal probabilities of 1-year olds: The probability of dispersing from a good to a poor quality habitat is p, while q is the probability of dispersing from the poor to the good habitat. This is also an example of source-sink dynamics, where the good habitat (the source) is sustaining the population living in the poor habitat (a sink). There is a limited number available number of breeding spots in good habitats, so some individuals are forced to use a lower quality site.

The survival coefficients are given by $\mathbf{s} = [0.8, 0.82, 0.82, 0.82, 0.82]$ for both sites, while the fertility coefficients differ. In the good site they are given by $\mathbf{f}_{\text{good}} = [0, 0.096, 0.16, 0.224, 0.32]$, and in the poor site by $\mathbf{f}_{\text{poor}} = [0, 0.1, 0.16, 0.2, 0.2]$. Thus, for the youngest breeders the fertility is slightly higher in the poor site, but for older breeders the fertility is higher in the good site.

In R we can construct the projection matrix as a function of the dispersal rates p and q as follows:

```
Sboth <- c(.8, rep(.82,4))
Fgood <- c(0, 0.096, 0.16, 0.224, 0.32)
Fpoor <- c(0, 0.1, 0.16, 0.2, 0.2)

AmatGulls <- function(p, q){
  #Survival in good environment:
  SGood <- c((1-p)*Sboth[1], Sboth[2:5])
  #Survival in poor environment:
  SPoor <- c((1-q)*Sboth[1], Sboth[2:5])
  Amat <- matrix(0, 10, 10)
  #-----------------------------------------------
  # Fertility in good site:
  Amat[1, 1:5] <- Fgood
  # Fertility in poor site:
  Amat[6, 6:10] <- Fpoor
  #-----------------------------------------------
  #P(survive and remain), good site
```

```
  diag(Amat[2:5,1:4]) <- SGood[1:4]
  #P(survive), last class
  Amat[5,5] <- SGood[5]
  #P(survive and remain), poor site
  diag(Amat[7:10,6:9]) <- SPoor[1:4]
  #P(survive), last class
  Amat[10,10] <- SPoor[5]
  #-----------------------------------------------
  #Survive / disperse from good to poor:
  Amat[7,1] <- p*Sboth[1]
  #Survive / disperse from poor to good
  Amat[2,6] <- q*Sboth[1]
  #-----------------------------------------------
  Amat
}
```

With this function we can make calculate λ for different combinations of the two dispersal probabilities and plot the result. Here we also use the `expand_grid` function, which is useful for making new data frames for combinations of variables:

```
#Sequences of p and values
pVec <-  seq(0,1,.01)
qVec <-  seq(0,1,.01)

gulls_table <- expand_grid("p"=pVec, "q"=qVec )

lambdagulls <- function(p, q){
  uvlambda(AmatGulls(p=p,q=q))$lam
}

gulls_table <- gulls_table %>% rowwise() %>%
  mutate(lambda=lambdagulls(p,q))

ggplot(gulls_table, aes(x=p, y=q, fill=lambda) ) +
  geom_tile()+
  scale_fill_viridis_c()  +
  theme_classic()+
  labs(fill=expression(lambda))+
  theme(legend.position = "top" )
```

4.4.3 Disease models

Stage structured matrix model play an important role in understanding disease dynamics, for systems that can be analysed with discrete time models. Classi-

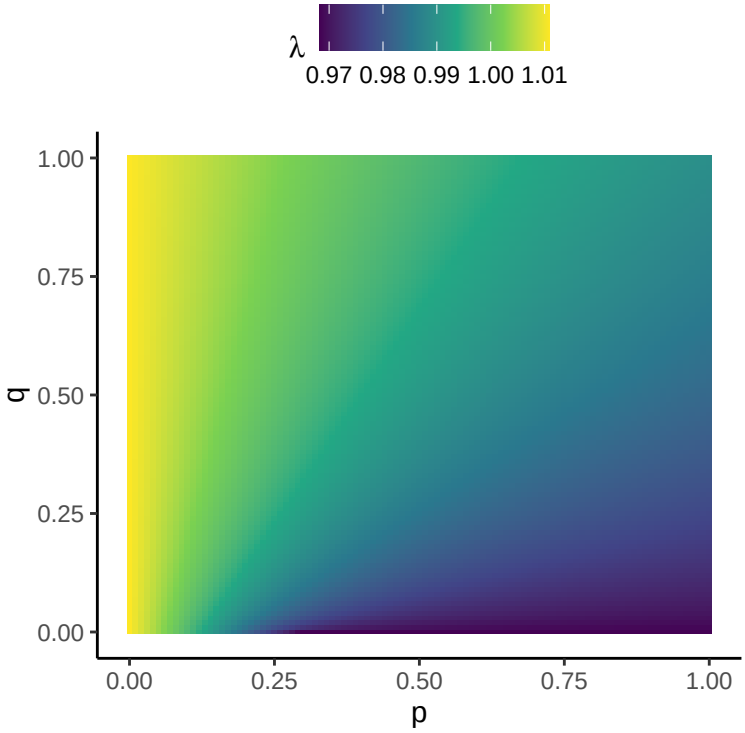


Figure 4.4: Surface of λ as a function of dispersal parameters p (probability of leaving the 'good' site) and q (probability of leaving the 'poor' site) in the black-headed gull model.

fying host individuals according to disease status (susceptible, infected, etc) is essential, and often other sources of variation need to be accounted for as well (age, sex, and other factors that can affect disease transmission and immunity). Space is another fundamental factor that often needs to be included to accurately capture the dynamics of an epidemic. In host-vector-pathogen systems the vector can also be classified according to multiple stages.

Disease dynamics are inherently frequency-dependent, so cannot be analyzed with the density independent methods presented so far, but they often have equilibrium dynamics that are analysed with matrix model methods. The parameter R_0 (that we now know as the lifetime reproductive success) has an important interpretation in disease models as the total number of new infections that each infected individual will give rise to in a susceptible population. In other words it is the lifetime reproductive success of the pathogen.

One of the simplest and most common models for disease dynamics is the SIR model [Kermack and McKendrick, 1927], classifying host individuals into three disease states (Susceptible - Infected - Recovered). This is an example of a compartmental model in epidemiology.

4.5 Integral projection models (IPMs)

Many state variables are intrinsically continuous, such as body size. Matrix models can still be used to study size structure by lumping individuals into different discrete size classes, but this adds an extra modeling decision of where to set the limits for each size class. And sometimes this decision affects the results of the model. This can be avoided by using an IPM, i.e. an **integral projection model** [Easterling et al., 2000], which accomodates continuous state variables. Instead of a population vector with a discrete number of stages, IPMs track individuals over a continuous trait distribution $n(z)$, where z represents the trait measure (e.g. size), and instead of a projection matrix, IPMs use a projection kernel. You can think of an IPM as a matrix model with infinitely many stages, and in fact they belong to the same class of models (matrix models). The good news is that IPMs can be analysed with matrix model methods (e.g. those introduced in chapter 3) to find λ and other model outputs. In R, we then specify the mesh size of the kernel (creating a matrix), and the finer the mesh size the more accurate are the estimated model outputs.

IPMs and matrix models are both part of the same model framework (matrix models). The main difference between an IPM and a matrix model is in how the underlying parameters are (typically) estimated from data: With IPMs we typically use some kind of regression model, where the state variable is an explanatory variable explaining the survival, fecundity and transition rates that define the model. With matrix models, these parameters are usually estimated independently within each stage, without treating the stage as an explanatory

variable - however regression models can still be used to estimate parameters of a matrix model. We will not go into details regarding IPMs here, but you have already learned the methods for analyzing them, as they are the same as for matrix models. For more details on IPMs and their application, see Ellner et al. [2016].

4.6 Exercises

For suggested solutions, see appendix A.

Exercise 4.1

1. Draw the life cycle graph for each of the projection matrices below.

$$\mathbf{A} = \begin{bmatrix} 0.5 & 2 \\ 1 & 0.7 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 10 \\ 0.9 & 0 & 0 \\ 0 & 0.9 & 0 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 8 & 5 \\ 0.9 & 0.1 & 0.7 \\ 0 & 0.7 & 0.2 \end{bmatrix}$$

2. Can you determine the stage-specific survival probability in each case? What assumption(s) do you need to make in order to do this?
3. What could the stages represent in each case?

Exercise 4.2

This exercise is based on an experimental toxicology study by Kammenga et al. [2001], and their model for the isopod common woodlouse (*Porcellio scaber*) was also used as an example by Vindenes et al. [2020]. The woodlouse model has seven discrete life history stages and monthly time steps.

There are 7 main stages in this woodlouse model:

1. Oocytes
2. Eggs/offspring (in marsupium)

Figure 4.5: Common woodlouse (*Porcellio scaber*).

3. Juveniles
4. First (spring) reproductive stage
5. Non-reproductive adult stage
6. Second (summer) reproductive stage
7. Senescent non-reproductive stage

The following life cycle graph is from Kammenga et al. [2001] (their fig. 1B):

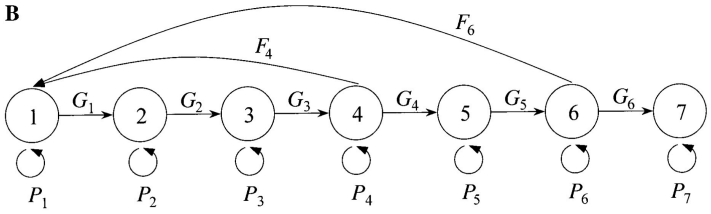


Figure 4.6: Life cycle graph for the common woodlouse (*Porcellio scaber*).

The vectors of P, G and F in R have the following values:

Stage	P	G	F
1	0.158	0.722	0.000
2	0.545	0.245	0.000
3	0.539	0.231	0.000
4	0.162	0.788	2.355
5	0.008	0.792	0.000
6	0.136	0.664	5.568
7	0.800	0.000	0.000

1. Use the life cycle graph and the vectors defined in the table to define the fertility matrix **F**, the survival matrix **U**, and the projection matrix **A** in R.

2. Decompose the matrices **U** and **F** to calculate the stage-specific fertility and survival probability, and the transition matrix (based on the decomposition in equation (4.1). Plot the stage-specific fertility and survival probability.

3. Calculate λ, the stable stage structure **u** and reproductive values **v**, and plot **u** and **v**. How fast does this population grow per year? Does this model work well to describe a life history that reproduces twice per year [Kammenga et al., 2001]?

4. Write an R function that returns the value of λ for this population for a set of three vectors corresponding to *proportional changes* to elements of P, G or F. What happens to λ, **u** and **v** if you modify the element G_6 in the vector G? Explain why.

5. What happens to λ, **u** and **v** if you modify the element G_4? Explain why.

Exercise 4.3

This exercise will continue with the woodlouse model from the previous exercise.

Kammenga et al. [2001] considered toxicological effects of cadmium. At the highest concentration, the woodlouse will switch to only one reproductive event instead of two, and the survival probabilities are reduced between 7 and 12 % in all stages except the egg stage. In addition the development time of eggs decreased [Kammenga et al., 2001]. For this exercise we assume the values of P, G, and F corresponding to cadmium exposure are given by

Stage	P	G	F
1	0.158	0.722	0.000
2	0.491	0.220	0.000
3	0.485	0.208	0.000
4	0.145	0.710	2.355
5	0.007	0.713	0.000
6	0.122	0.598	0.000
7	0.720	0.000	0.000

1. Define a new matrix $\mathbf{A}_C$ (C for cadmium) in R based on these values. This is now a treatment matrix, while the previously defined matrix is the reference matrix.

2. Calculate λ for the new matrix. How does it compare to the previous value?

3. Calculate the halfway matrix $\mathbf{A}^*$ and the difference matrix **D**.

4. Calculate the sensitivity matrix $\mathbf{S}^*$ based on the halfway matrix, and finally calculate and plot the elements of the matrix corresponding to the

contributions to the difference in λ between $\mathbf{A}$ and $\mathbf{A}_C$. Which element contributes the most?

5. Why is it often better to use an LTRE than a sensitivity analysis when we want to understand the impacts of an external factor such as a toxicant?

Chapter 5

Stochasticity

This chapter will give an introduction to stochastic environments and bet-hedging strategies. The underlying theory is based on stochastic processes and probability theory, and this chapter will only touch upon a few basic concepts from this large research field. A key result from the most basic model of stochastic population growth is that the long-term population growth rate (fitness) in variable environments will always be lower than the growth rate in the constant mean environment.

5.1 Learning goals

- Understand key properties of population growth in variable environments.
- Describe the main properties of the Lewontin-Cohen model.
- Understand how bet-hedging strategies can help individuals live in variable environments.
- Discuss the role of variable and unpredictable environments in light of climate change.

5.2 Variable vs unpredictable environments

So far we have only considered populations in constant environments. But natural environments can be highly variable, both in time and space. Here, 'environment' is a very general term which can refer to any particular abiotic variable (like temperature) or a combination of several factors that affect organisms living in the habitat, including biotic factors like resource availability. Environmental variability can be *predictable*, like the seasonal changes in light

conditions at high latitudes, or *unpredictable* - which is the case for stochastic environments. Unpredictable variation also includes rare events such as rainfalls in a dessert, or flooding and wildfires. We can estimate a frequency at which such events typically occur over many years, but it is difficult to predict the exact timing of each such event.

Practically all organisms live in some kind of variable environment (although some habitats are more stable than others, for instance the deep depths of the oceans), and species have evolved various strategies to deal with both the wide range of possible conditions they (or their offspring) may encounter, under varying degrees of predictability. These adaptations allow individuals to mitigate negative impacts of bad conditions, as well as take advantage of good conditions, and spread the risk associated with uncertainty in various ways.

If the environment is fluctuating in a predictable way, **phenotypic plasticity** can allow individuals to cope with the variation. Organisms can then take advantage of environmental cues during development to achieve an optimum phenotype in the future environment. For unpredictable (stochastic) environments, however, organisms cannot rely on such cues. In these cases **bet-hedging** strategies may evolve.

5.3 The Lewontin-Cohen model

The simplest model for stochastic population growth applies to an unstructured population. In this model the annual population growth rate is a stochastic variable Λ_t with mean $\mathrm{E}[\Lambda_t] = \lambda$, variance $\mathrm{Var}(\Lambda) = \sigma^2$, and with no autocorrelation between the growth rates of different years. This means that the annual growth rates are IID (Identically and Independently Distributed), and this assumption is important for the derivation of the long-term population growth rate. This model is the Lewontin-Cohen model [Lewontin and Cohen, 1969], also known as a 'geometric random walk'.

5.3.1 Defining the process

Starting from a population size N_0, the population size after t time steps in the Lewontin-Cohen model is given by

$$N_{t+1} = \Lambda_t \Lambda_{t-1} \cdots \Lambda_1 \Lambda_0 N_0 = N_0 \prod_{i=0}^{t} \Lambda_i. \tag{5.1}$$

We are interested in the behavior of a 'typical' population growing according to this process, and to define the long-term growth rate of such a 'typical' population (the socalled stochastic growth rate). As we will see below, this is not

the growth rate of the mean population size, but rather the median population size. Before we go into details of this derivation, we will look at how we can generate realizations of the Lewontin-Cohen process in R and plot one or more realizations.

Plot one realization of the process

The R function `LC` below can be used to generate a random sequence of population sizes over time based on this model (each time you run the code it will generate a different sequence, unless you use the function `set.seed()` which initializes the pseudorandom number generator in R). The `LC` function returns the sequence of population sizes as a data frame, including the time steps from 0 to the set number of years. The output from one run of the function is shown in figure 5.1. In this example, a normal distribution is used to generate the sequence of annual growth rates, but the exact kind of distribution is not critical as long as the mean and variance can be set independently (for instance, a lognormal distribution could be used since all the multiplicative growth rates should be non-negative).

```
#mean.lambda: Expectation of annual lambda
#sd.lambda: Standard deviation of lambda

LC <- function(Tmax=30,
               n0=100,
               mean.lambda=1.05,
               sd.lambda=0.1){
  stochastic.lambdas <- rnorm(Tmax,
                              mean=mean.lambda,
                              sd=sd.lambda)
  Nvec <- rep(NA, Tmax+1)
  Nvec[1] <- n0
  for(i in 1:Tmax){
    Nvec[i+1] <- stochastic.lambdas[i]*Nvec[i]
  }
  data.frame("Time"=0:Tmax, "N"=Nvec)
}

#Initialize R's random number generator,
#the number chosen is arbitrary
set.seed(20)

#Apply function and plot result:
StochasticGrowth <- LC()

ggplot(StochasticGrowth) +
```

```
  geom_line(aes(x=Time, y=N), lwd=1.2)+
  geom_point(aes(x=Time, y=N), size=3)+
  theme_bw() +
  labs( x="Time (year)", y="Population size N")
```

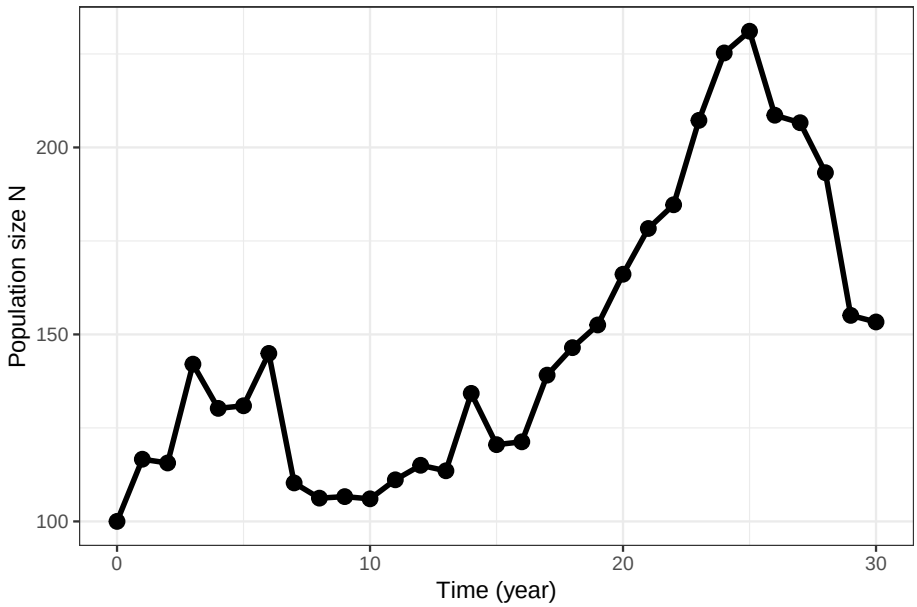


Figure 5.1: Stochastic population growth following the Lewontin-Cohen model with normally distributed annual growth rates with mean $\lambda = 1.05$ and standard deviation $\sigma = .1$

This code only generates one realization of the growth process. To learn more about the typical behavior of this stochastic growth process we need to generate many realizations (from the same starting point, with the same mean and variance of the annual growth rate) and look at some summary statistics.

Plot many realizations of the process

The following function generates a chosen number (`nsim`) of realizations of the Lewontin-Cohen model, stores them in a data frame and makes a plot of all of them.

```
nsimLC <- function(nsim=100, Tmax=30, n0=100,
                   mean.lambda=1.05, sd.lambda=0.1){
  frame <-  data.frame("Time"=0:Tmax)
  for(j in 1:nsim){
  stochastic.lambdas <- rnorm(Tmax, mean=mean.lambda, sd=sd.lambda)
```

```
  Nvec <- rep(NA, Tmax+1)
  Nvec[1] <- n0
  for(i in 1:Tmax){
    Nvec[i+1] <- stochastic.lambdas[i]*Nvec[i]
  }
  frame <- cbind(frame, Nvec)
  names(frame)[j+1] <- paste("N",j,sep="")
  }
  frame
}
```

To apply this function and plot the results, we can use the following code:

```
set.seed(20)

simLC100 <- nsimLC()

simLC100.long <- simLC100 %>% pivot_longer(c(-Time),
               names_to = "Rep", values_to = "N")

ggplot(simLC100.long) +
  geom_line(aes(x=Time, y=N, col=Rep), lwd=.5)+
  scale_color_manual(values=rep(1,100))+
  theme_bw() +
   theme(legend.position = "none" )+
  labs( x="Time (year)", y="Population size N")
```

Figure 5.2 shows the output from 100 realizations. We see that over time, these realizations diverge more and more from each other as the randomness accumulates. A few realizations reach large population sizes, while many are located in intermediate values. However, it quickly gets quite messy to plot so many realizations, and if we want more than 100 it will be even worse. If we want to summarize the outcome of many realizations over time, a better option is to plot selected **percentiles** over time.

Plot percentiles from many realizations

Percentiles (or quantiles) are useful for summarizing outputs from many realizations. For instance, the 50%%% percentile represents the population size at which 50% of the realizations will lie above it and 50% below it (i.e. the median in the limit as the number of realizations approaches infinty), the 95% percentile represents the population size at which 5% of the realizations lies above it and 5% below it, etc. The following code makes such a plot based on 10 000 realizations of the Lewontin-Cohen process. It can take some seconds to run (reduce the number of realizations `nsim` if it gets too slow)

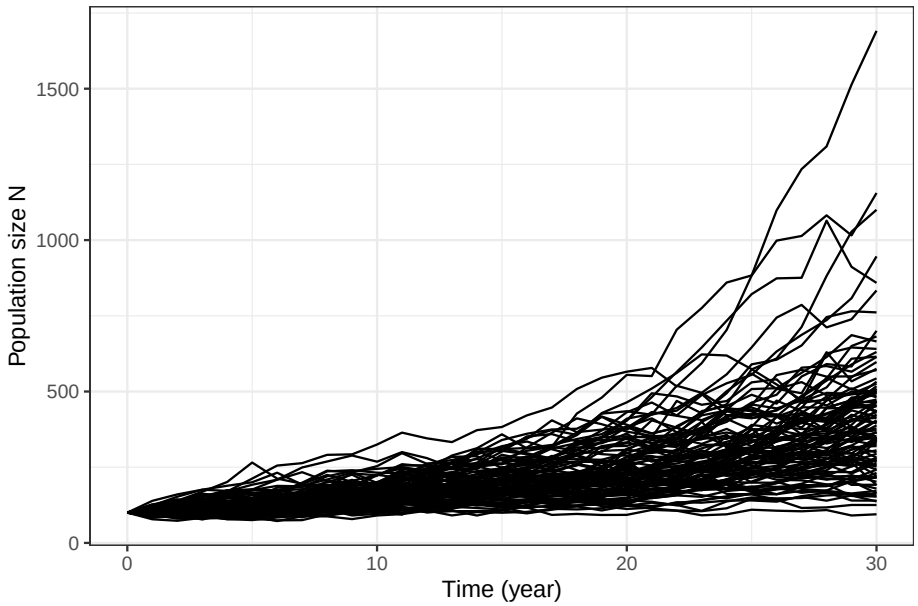


Figure 5.2: 100 realizations of the Lewontin-Cohen model with normally distributed annual growth rates, with mean $\lambda = 1.05$ and standard deviation $\sigma = .1$

```
#Apply function to get 10 0000 realizations (can take some time)
simLC_Large <- nsimLC(nsim=10000)

#Get the 5, 25, 50, 75, and 95 percentiles:
Percentiles1  <- t(apply(t(simLC_Large[,-1]),
                         2,
                         quantile,
                         probs=c(.05, .25, .5, .75,.95)))


#Store percentiles in data frame
LCPercentiles <- as.data.frame(cbind("Time"=0:(dim(Percentiles1)[1]-1),
                                     Percentiles1))

LCPercentiles$Mean <- apply(t(simLC_Large[,-1]),2, mean)

#Make long format for plotting
LCPercentiles.long <- LCPercentiles %>% pivot_longer(c(-Time),
               names_to = "Percentile", values_to = "N")

#Make sure the order of the percentiles is correct
```

```
LCPercentiles.long$Percentile <- factor(
  LCPercentiles.long$Percentile,
  levels=c("5%", "25%", "50%", "75%", "95%","Mean"))

ggplot(LCPercentiles.long) +
  geom_line(aes(x=Time, y=N, col=Percentile,
                linetype=Percentile), lwd=1)+
  scale_color_manual(values=c(colors5,2))+
  scale_linetype_manual(values=c(rep(1,5),2))+
  theme_bw() +
   theme(legend.position = "top",
         legend.title = element_blank())+
  labs( x="Time (year)", y="Population size N")+
  guides(color = guide_legend(nrow = 1, byrow = TRUE))
```

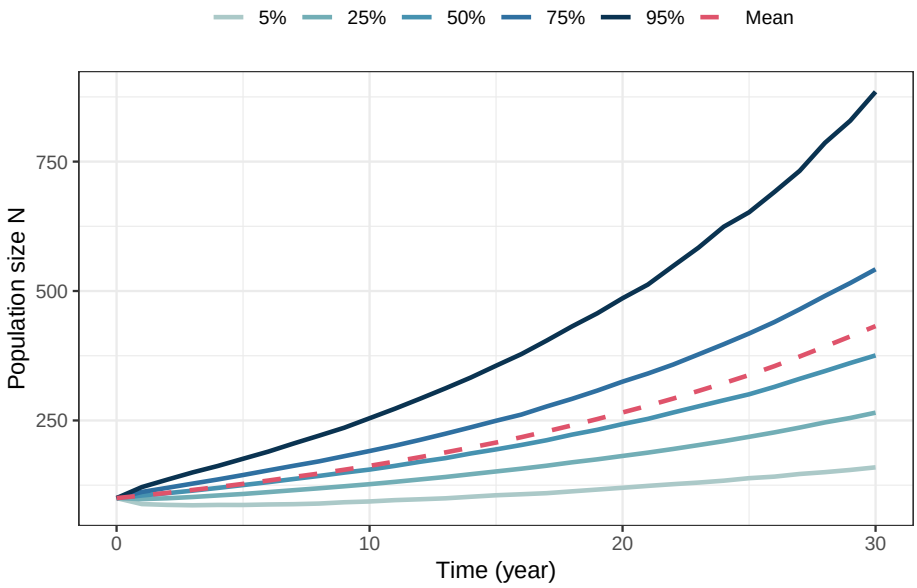


Figure 5.3: Percentiles and mean from 10000 simulated trajectories of the Lewontin-Cohen model with normally distributed annual growth rates with mean $\lambda = 1.05$ and standard deviation $\sigma = 0.1$.

In contrast to the realizations, the percentiles do not fluctuate (the higher the number of realizations, the more smooth they will be). The 50% percentile represents the median and is therefore a good representation of the 'typical' population in the stochastic environment. The mean population size (shown in red on figure 5.3) is not a good measure of the typical population, because only only one or a few realizations reaching very high values can have a large impact on this value. Thus, for stochastic growth models the long-term growth

rate of the median population is a better fitness measure than the long-term growth rate of the mean population. In extreme cases when the variance of the annual growth rates is large, the expected population size can increase exponentially because a few realizations reach extremely high values, while the median population size is declining and each realization will eventually go extinct with probability 1. We will get back to this below.

<br

5.3.2 The stochastic growth rate

First, we consider the growth rate of the mean population size, although we already know this is not the best measure for the typical population. Because we assumed that the Λ_t's are IID (see above), the expected population size after t time steps is given by

$$\mathrm{E}[N_t] = N_0 \mathrm{E}\left[\prod_{i=0}^{t-1} \Lambda_i\right] = N_0 \prod_{i=0}^{t-1} \mathrm{E}[\Lambda_i] = N_0 \lambda^t. \tag{5.2}$$

Thus, the expected population size grows at the rate λ (the mean of the annual growth rate), just like a population growing exponentially in a constant environment. Note that λ does not depend on the variance of the annual growth rates.

To derive the long-term growth of the 'typical' population (the socalled **stochastic growth rate**) we look at the Lewontin-Cohen process (equation (5.1)) on log scale:

$$\ln N_t = \ln \Lambda_{t-1} + \ln \Lambda_{t-2} + ... + \ln \Lambda_1 + \ln \Lambda_0 + \ln N_0 \tag{5.3}$$

$$= \ln N_0 + \sum_{i=0}^{t-1} \ln \Lambda_i. \tag{5.4}$$

Here, the first term $\ln N_0$ is a constant (since the initial population size is given). The second term is a sum over all the stochastic growth increments on log scale. Together with the IID assumption, this allows us to take advantage of the central limit theorem (if you need a reminder: https://en.wikipedia.org/wiki/Central_limit_theorem): As as the number of time steps t approaches infinity the sum will be normally distributed with mean $t\mathrm{E}[\ln \Lambda_t]$ and variance $t\mathrm{Var}(\ln \Lambda_t)$.

Thus, the expected value of $\ln N_t$ is given by

$$\mathrm{E}[\ln N_t] = \ln N_0 + \mathrm{E}\left[\sum_{i=0}^{t-1} \ln \Lambda_i\right] = \ln N_0 + t\mathrm{E}[\ln \Lambda_t] \tag{5.5}$$

This expectation grows (or declines) linearly over time t, and the slope of this line is the mean $\mathrm{E}[\ln \Lambda_t] = r_s$. The constant r_s is called the **stochastic growth rate** (note that the name can be misleading, as this is a constant, and not a stochastic variable).

Importantly, the stochastic growth rate is generally lower than the growth rate of the mean population, $r = \ln \lambda = \ln(\mathrm{E}[\Lambda_t])$, unless the variance of the annual growth rates is zero (then they are the same). The stochastic growth rate describes the growth of the median population size. The population size N_t is lognormally distributed because $\ln N_t$ is lognormally distributed (due to the central limit theorem), and the lognormal distribution is skewed. In the lognormal distribution median is always lower than (or equal to) the mean, and is a better descriptor of the *central tendency* (or as we put it, it provides a better description of a typical population growth trajectory than the mean).

How different are the two growth rates, and can we express this difference as a function of the variance of the Λ_t, σ^2? Yes, Lewontin and Cohen [1969] showed (using a Taylor expansion of $\ln \Lambda_t$ around λ; not within the scope of this compendium) that the stochastic growth rate r_S can be written as

$$r_s \approx r - \frac{\sigma^2}{2\lambda^2}. \tag{5.6}$$

This equation tells us that the difference between the growth rate r in the constant environment and r_s in the stochastic environment is increasing with the value of σ^2. Thus, we can have situations where the population growth rate r is positive, but r_S is negative. And in that case, extinction of the population (at some point) is certain.

5.3.3 Arithmetic and geometric mean

The stochastic growth rate $r_s = \mathrm{E}[\ln \Lambda_t]$ and the growth rate of the mean environment $r = \ln \mathrm{E}[\Lambda_t])$ can also be compared by considering the arithmetic and geometric mean of a series of annual growth rates Λ_t. The arithmetic mean is given by

$$\lambda = \mathrm{E}[\Lambda_t] = \frac{1}{t}\sum_{i=1}^{t} \Lambda_i, \tag{5.7}$$

and the log of this value is $r = \ln \lambda$. The geometric mean is given by

$$e^{\mathrm{E}[\ln \Lambda_t]} = \left(\prod_{i=1}^{t} \Lambda_i \right)^{1/t}, \tag{5.8}$$

and the log of this value corresponds to $r_S = \mathrm{E}[\ln \Lambda_t]$. In contrast to the arithmetic mean, the geometric mean depends on the variance σ^2. The arithmetic mean will always be greater than or equal to the geometric mean, and in this case they are only equal in the constant environment (if $\sigma^2 = 0$).

The code below returns the arithmetic and geometric mean of a sequence of annual growth rates, and plots them along with the sequence:

```
AGMeans <- function(Tmax=100, mean.lambda=1.05, sd.lambda=.3){
  #Use lognormal distribution to avoid negative values
  Lambdavec <- rlnorm(n=Tmax,
                      log(mean.lambda)-
                        .5*log(1+ sd.lambda^2/
                                 (mean.lambda^2)),
                      sdlog=sqrt(log(1+sd.lambda^2/
                                       (mean.lambda^2))))
   geomLambda <- exp(mean(log(Lambdavec)))
   meanLambda <- mean(Lambdavec)
  data.frame("Time"=0:(length(Lambdavec)-1),
             "Lambda"=Lambdavec,
             "Geometric"=geomLambda,
             "Arithmetic"=meanLambda)
}
set.seed(20)

GetMeans <- AGMeans()
Geometric <- GetMeans$Geometric[1]
Arithmetic <- GetMeans$Arithmetic[1]
ggplot(GetMeans) +
  geom_point(aes(x=Time, y=Lambda), size=2, col=8)+
  geom_hline(yintercept= Geometric,lwd=1.2,
             linetype=2,col=2)+
  geom_text(aes(10, Geometric,
                label = "Geometric mean",
                vjust=2), col=2)+
  geom_hline(yintercept=Arithmetic,lwd=1.2,
             col=4)+
  geom_text(aes(10,  Arithmetic,
                label = "Arithmetic mean",
                vjust=-2),col=4)+
```

```
  theme_bw() +
  theme(legend.position = "none" )+
  labs( x="Time (year)", y=expression(Lambda[t]))
```

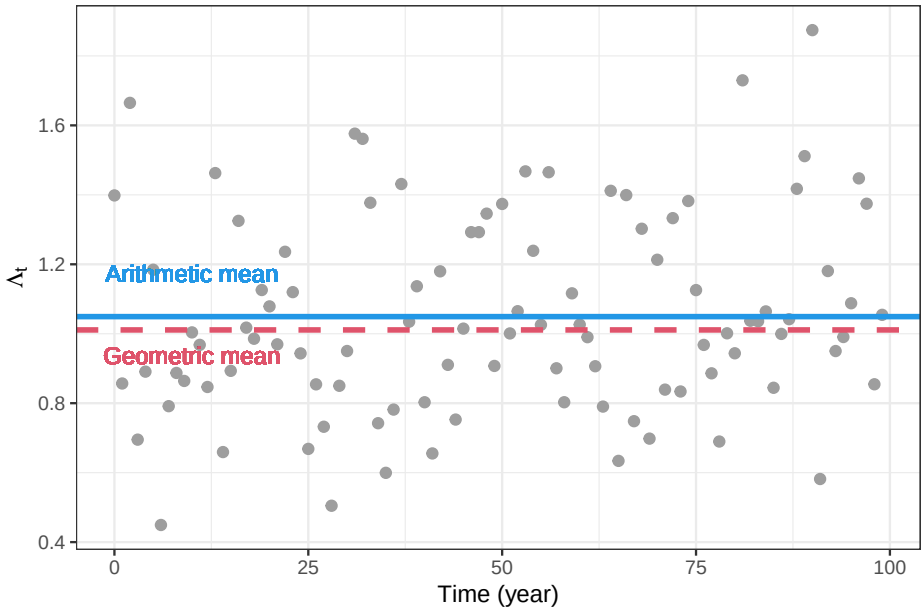


Figure 5.4: Arithmetic and geometric mean of a sequence of annual growth rates Λ_t, with annual mean $\lambda = 1.05$ and annual standard deviation $\sigma = 0.3$

5.3.4 Bet-hedging

From an evolutionary perspective, the Lewontin-Cohen model is an important starting point to describe fitness in a stochastic environment (r_S), and understand how it differs from the fitness in a the mean environment (r). In particular, it provides a baseline model to describe strategies of bet-hedging. Since r_S depends on both r and σ^2, organisms in stochastic environments can improve their fitness through increasing the mean r, or reducing the variance σ^2. **Bet-hedging** strategies *reduce the variance, at the cost of also reducing the arithmetic mean fitness*. As a result the fitness in the stochastic environment r_S is optimized.

The study of bet-hedging started with the work of Cohen [1966] to understand the fitness of plants in unpredictable environments like a desert. Bet-hedging can operate through two main mechanisms, or a combination [Starrfelt and Kokko, 2012]:

- **Conservative bet hedging** refers to strategies that 'play it safe', where the organism does not optimize its fitness for any given environment, but in the long run still come out with a higher geometric mean fitness. For instance, a bird could lay a low number of eggs at each reproduction to improve the survival of their offspring, thus reducing the variance but at the cost of a lower than optimum clutch size in the mean environment.
- **Diversifying bet-hedging** are strategies that avoid risk by 'not putting all eggs in one basket', for instance by producing offspring with a high diversity of phenotypes. While this will certainly increase the variance in fitness among the offspring, it reduces the variance of the parent. Again, it is not truly bet-hedging unless the parent also pays a cost in terms of lowered arithmetic mean fitness (here corresponding to producing offspring of just one type optimal for the mean environment).

5.4 Stochastic growth of structured populations

In structured populations, the stochastic population dynamics are more complex because the annual population growth rates are no longer 'IID' (Independently and Identically Distributed). For the Lewontin-Cohen model this was a key assumption that allowed us to use the central limit theorem to derive the stochastic growth rate. But in structured populations, the annual growth rates will show *autocorrelation* (a memory in the process) because the population structure itself generates lags in the growth process. Even if there is some stochastic environmental variable that is IID and gives rise to stochasticity in the population growth, the population growth rates will not be IID.

For instance, if the environment is particularly good for reproduction one year, this can lead to a 'strong' year class (cohort) that persists in the population over time and affects the population growth rate for potentially many years. We get 'waves' of such cohorts that generate long-term fluctuations in the stage structure on top of the year-to-year fluctuations due to stochasticity. The long-term fluctuations create autocorrelation in the population growth process. In contrast to the population growth in a constant environment, where the transient initial fluctuations disappear over time (chapter 3), the transients will never really disappear in a stochastic environment Instead, the population structure will fluctuate around the stable structure.

How can we then study population growth and fitness of structured populations in stochastic environments? We can always do simulations (projections) of the population growth process, as we did for the Lewontin-Cohen model, and we can define the stochastic growth rate based on a (long) sequence of generated populaton sizes. In addition, under the assumption of small fluctuations it is possible to approximate the stochastic growth rate as a function of the mean, variance and covariance of matrix elements, as we will see below. First we will

define a general model to describe the growth of structured populations in a time-varying environment.

5.4.1 Defining the process

In a fluctuating environment, the projection matrix $\mathbf{A}$ will take a new value each time step, so that the population growth over many time steps is given by matrix multiplication from time 0:

$$\mathbf{n}_{t+1} = \mathbf{A}_t\mathbf{n}_t = \mathbf{A}_t\mathbf{A}_{t-1}\mathbf{A}_{t-2}\cdots\mathbf{A}_0\mathbf{n}_t$$

When these time-dependent projection matrices $\mathbf{A}_t$ are generated by a random process each time step, we have a **stochastic matrix model**. There are two main approaches to study the dynamics of such a model:

- The *environment-blind* approach generates a set of projection matrices e.g. from each year of a study, and then samples among these to generate a new stochastic sequence of matrices according to a given sampling model (e.g. a Markov process). In this case, correlation between vital rates of the projection matrix are automatically incorporated since the entire matrix is sampled.
- The *environment-specific* approach specifies each matrix element (e.g. fertility and survival coefficients) as a stochastic variable with a mean, variance and potentially covariance among matrix elements. Together, all the stochastic elements in the model define a multivariate vector with a given distribution (specified by a vector of means and a variance-covariance matrix). This approach is more complex, but often better suited to account for explicit relationships between the vital rates and environmental variables such as temperature.

Bird example

We will again use the bird example introduced in @ref{birdtable} and following chapters, and now consider a stochastic version of the matrix model for this bird with pre-reproductive census (chapter 3).

In the constant environment, the pre-reproductive projection matrix for the songbird was given by ((3.6)):

$$\mathbf{A} = \begin{bmatrix} f_1 & f_2 & f_3 \\ s_1 & 0 & 0 \\ 0 & s_2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.6 & 1.2 \\ 0.9 & 0 & 0 \\ 0 & 0.6 & 0 \end{bmatrix}. \tag{5.9}$$

We will now add stochasticity to the elements of this matrix, by allowing the fertilities and survival probabilities to fluctuate randomly. Adding stochasticity needs to be done with some care: Survival probabilities are restricted to be between 0 and 1, and fertilities must always be non-negative. A good idea is to use appropriate *scaling functions* (link functions) where stochasticity can be added as a normally distributed variable on the link scale. For this example, we will assume a log-link function for the fertilities, and a log log link function for the survival (but in general many other kinds of functions are also possible):

$$f_{it} = \exp(\beta_{fi} + \varepsilon_{fit})$$

$$s_{it} = \exp(-\exp(-\beta_{si} - \varepsilon_{sit}))).$$

Here ε_{fi} and ε_{si} are random (stochastic) variables assumed to follow a normal distribution with mean 0 and standard deviation σ_{fi} and σ_{si}, respectively. The constants β_{fi} and β_{si} define the fertility and survival when the standard deviations (and thus the ε's) are zero. Note that this is just one of several ways to model stochasticity in survival and fertility.

Plot one realization of the process

To add stochasticity to the fertilities and survival probabilities in the projection matrix, we will use the function `mvrnorm()` from the package `MASS`, which generates values from the multivariate normal distribution. The function `mvrnorm` uses a variance-covariance matrix as one of its input variables, and here we will for simplicity assume no covariance. We also assume that stochasticity only affects survival and fertility coefficients, and that these follow log-log link and log-link functions, respectively.

The following function generates a series of random projection matrices using the `mvrnorm()` function from the package `MASS` to add stochasticity in survival and fecundity. The matrices are returned as an array, which is an extension of the ´matrix´ object to multiple dimensions. Here we have three dimensions: Time (`Tmax`), k and k, where k is the number of rows and columns of the projection matrix.

Inputs to the function are the projection matrix in the constant environment **A**, its main components **U** and **F** (used to separate out survival coefficients and fertilities), and two vectors defining the variance in scaled fertility and survival. Each vector needs to be of length k corresponding to the number of stages in the population.

```
Amat.array <- function(MatA,
                       MatU,
                       MatF,
```

```
                        tmax=30,
                        VarF=rep(0.03, 3),
                        VarS=rep(0.03, 3)){
  #Split survival/transition matrix:
  SplitU <- DecomposeU(MatU)
  #Split fertility matrix:
  SplitF <- DecomposeF(MatF)
   #Transition matrix:
  Gmat <- SplitU$Gmat
   #Survival vector:
  Svec <- SplitU$Survival
  #Define beta (log-log link function):
  Beta.S <- -log(-log(Svec))
  #Offspring transition matrix:
  Qmat <- SplitF$Qmat
  #Fertility vector:
  Fvec <- SplitF$Fertility
  #Define beta (log link function):
  Beta.F <- log(Fvec)

  #variance covariance matrix
  SigMat <- diag(c(VarS, VarF))
  #Draw random (scaled) values for S and F:
  SFTime <- mvrnorm(tmax, mu=c(Beta.S, Beta.F), Sigma=SigMat)
  #Number of matrix classes
  k <- length(Svec)
  #Rescaled survival probabilities (loglog link):
  SvecTime<-   exp(-exp(-SFTime[,1:k]))
  #Rescaled fertilities  (log link):
  FvecTime<- exp(SFTime[,(k+1):(2*k)])
#Build  projection matrix for each time step:
  A.array <- array(NA,dim=c(k,k,tmax))
  for (i in 1:tmax){
    Smat <- diag(SvecTime[i,])
    Fmat <- diag(FvecTime[i,])
    A.array[,,i] <-   Gmat%*%Smat+Qmat%*%Fmat
  }
return("Amats" = A.array)
}
```

Using this function to generate a sequence of stochastic projection matrices, we can make another function to project the population growth over time. This projection is similar to the one we used for the constant environment (chapter 3), except that for each time step we now use a new different matrix due to stochasticity. The function below returns a data frame with the size of each

stage over time, including the initial sizes provided in `n0`.

```
projection.stochastic <-function(Amats, n0){
    Tmax <- dim(Amats)[3]
    k <- dim(Amats)[1]
    if(length(n0)!=k){
      warning("n0 should have length k
              corresponding to number of
              stages in the Amats object")
      }
    Nmat <- matrix(NA,nrow= Tmax+1, ncol=k)
    Nmat[1,] <- n0
  cnames <- paste("n", 1:k, sep="")
  timesteps <- 0:Tmax
    for(i in 2:(Tmax+1)){
        Nmat[i,] <- Amats[,,i-1] %*% Nmat[i-1,]
        }
  frame <- data.frame(timesteps,Nmat)
  colnames(frame) <- c("time", cnames)
  frame
}
```

Bird example

```
set.seed(20)

#Generate sequence of stochastic matrices:
Amats.bird <- Amat.array(MatA=Abird.pre,
                         MatU=Ubird.pre,
                         MatF=Fbird.pre,
                         tmax=20,
                         VarF=rep(0.03, 3),
                         VarS=rep(0.03, 3))

#Project growth:
stochastic.bird <- projection.stochastic(Amats = Amats.bird,
                                         n0=rep(10,3))

#Long format for plotting:
stochastic.bird.long <- stochastic.bird %>% pivot_longer(c(-time),
               names_to = "AgeClass", values_to = "Value")

#Plot:
ggplot(stochastic.bird.long) +
  geom_line(aes(x=time, y=Value, col=AgeClass,
```

```
                linetype=AgeClass), lwd=1.2)+
  geom_point(aes(x=time, y=Value, col=AgeClass), size=3)+
  theme_bw() +
  scale_color_manual(values=colors3)+
  scale_linetype_manual(values=c(1,2,3))+
  labs( x="Time (year)", y="Age class size")+
  theme(legend.position = "top" )
```

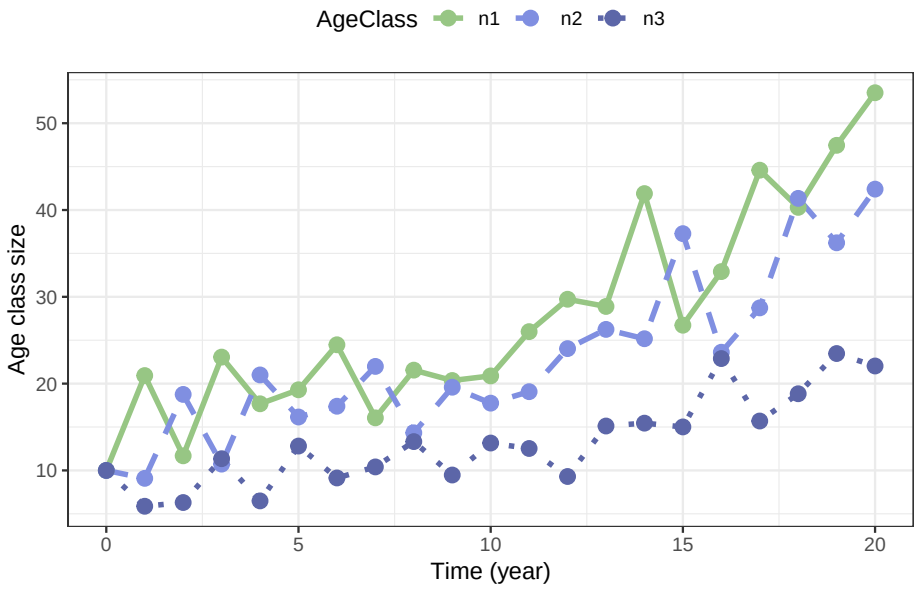


Figure 5.5: One realization of stochastic growth in the bird example with pre-reproductive census, with stochastic survival and fertility coefficients.

Figure **??**fig:birdstochastic1) shows the growth of each stage of the bird population over 20 years, starting from a population with 10 birds in each age class.The highest number of fluctuations is found for the first age class, and we can see that the peaks in this age class are followed by peaks in year class 2 and 3 in the following years, as the cohort grows through the population.

We can also plot the total population size, by summing up the values of the age classes for each year, as shown in figure 5.6:

```
#Sum up population sizes
stochastic.bird$Ntot <- apply(stochastic.bird[,-1],1,sum)

ggplot(stochastic.bird) +
  geom_line(aes(x=time, y=Ntot), lwd=1.2)+
  geom_point(aes(x=time, y=Ntot), size=3)+
  theme_bw() +
```

```
  labs( x="Time (year)", y="Populatipn size N")
```

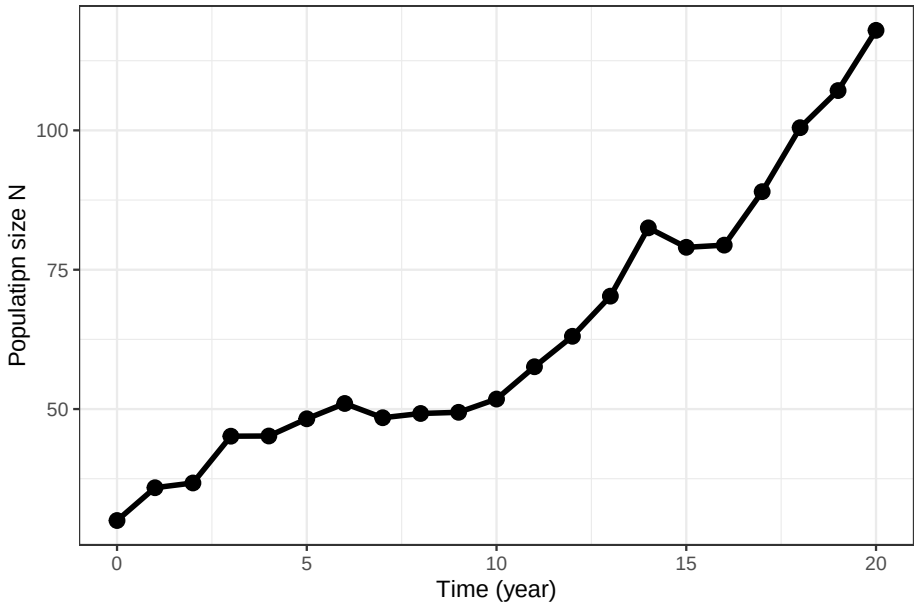


Figure 5.6: Same process as in figure 5.5, with age classes summed up to get the total population size.

Plot many realizations of the process

Instead of looking at just one realization of the stochastic growth process, it is often useful to generate multiple realizations and look at summary statistics and properties for these, as we have seen already for the Lewontin-Cohen-process.

The following function generates a series of matrices `nsim` times, each series of length `Tmax`, and returns the projected total population size over time for each series, in a data frame. Other input variables are the constant matrices and variance vectors, and initial population vector.

```
nsim.stochastic <- function(nsim=100,
                            tmax=30,
                            n0=rep(10,3),
                            MatA,
                            MatU,
                            MatF,
                            VarF,
                            VarS){
  frame <-  data.frame("Time"=0:tmax)
```

```
  for(m in 1:nsim){
  AMATS <- Amat.array(MatA=MatA,
                      MatU=MatU,
                      MatF=MatF,
                      VarF=VarF,
                      VarS=VarS,
                      tmax=tmax)
   projectN <- apply(projection.stochastic(Amats=AMATS,
                                           n0=n0)[,-1],1,sum)
  frame <- cbind(frame, projectN)
  names(frame)[m+1] <- paste("Rep",m,sep="")
  }
  frame
}
```

Bird example

We can now plot multiple realizations also in the bird example, as shown in figure 5.7

```
set.seed(20)

nsim.bird <- nsim.stochastic(MatA=Abird.pre,
                             MatU=Ubird.pre,
                             MatF=Fbird.pre,
                             VarF=rep(0.03, 3),
                             VarS=rep(0.03, 3),
                             tmax=20, nsim=100,
                             n0=rep(10,3))


nsim.bird.long <- nsim.bird %>% pivot_longer(c(-Time),
               names_to = "Rep", values_to = "N")

ggplot(nsim.bird.long) +
  geom_line(aes(x=Time, y=N, col=Rep), lwd=.5)+
  scale_color_manual(values=rep(1,100))+
  theme_bw() +
   theme(legend.position = "none" )+
  labs( x="Time (year)", y="Population size N")
```

To estimate the stochastic growth rate r_S from multiple realisations, we can use the following function (this measure is based on the mean of the annual growth increments of $\ln N$):

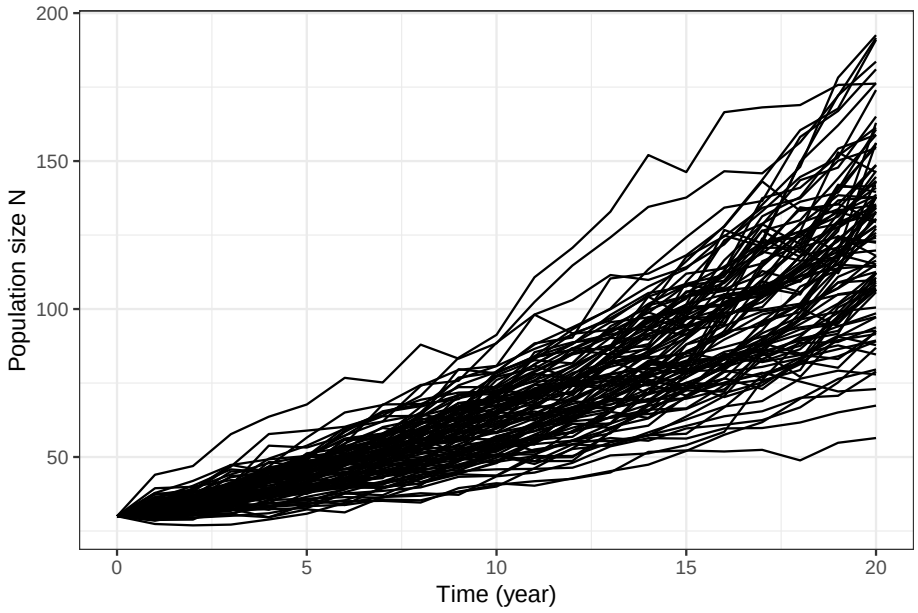


Figure 5.7: 100 realizations of the stochastic bird growth process with same parameters as in figure 5.6.

```
diffmat <- apply(log(nsim.bird[,-1]),2,diff)
rS <- mean(apply(diffmat,2,mean))
```

In this example, the stochastic growth rate is $r_S \approx 0.07$.

Plot percentiles from many realizations

As shown for the Lewontin-Cohen model we can also plot percentiles of the stochastic population trajectories from a structured model, after running multiple realizations.

Bird example

For the bird example with 20 time steps, we get

```
set.seed(20)

#Apply function to get 10 0000 realizations (can take some time)
simbird_Large <- nsim.stochastic(MatA=Abird.pre,
                                 MatU=Ubird.pre,
                                 MatF=Fbird.pre,
```

```
                                VarF=rep(0.03, 3),
                                VarS=rep(0.03, 3),
                                tmax=20,
                                nsim=10000,
                                n0=rep(10,3))

#Get the 5, 25, 50, 75, and 95 percentiles:
Percentiles2  <- t(apply(t(simbird_Large[,-1]),
                         2,
                         quantile,
                         probs=c(.05, .25, .5, .75,.95)))


#Store percentiles in data frame
BirdPercentiles <- as.data.frame(cbind("Time"=0:(dim(Percentiles2)[1]-1),
                                       Percentiles2))

BirdPercentiles$Mean <- apply(t(simbird_Large[,-1]),2, mean)

#Make long format for plotting
BirdPercentiles.long <- BirdPercentiles %>% pivot_longer(c(-Time),
               names_to = "Percentile", values_to = "N")

#Make sure the order of the percentiles is correct
BirdPercentiles.long$Percentile <- factor(BirdPercentiles.long$Percentile,
                    levels=c("5%", "25%", "50%", "75%", "95%","Mean"))

ggplot(BirdPercentiles.long) +
  geom_line(aes(x=Time, y=N, col=Percentile,
                linetype=Percentile), lwd=1)+
  scale_color_manual(values=c(colors5,2))+
   scale_linetype_manual(values=c(rep(1,5),2))+
  theme_bw() +
   theme(legend.position = "top",
         legend.title = element_blank())+
  labs( x="Time (year)", y="Population size N") +
   guides(color = guide_legend(nrow = 1, byrow = TRUE))
```

5.4.2 Stochastic growth rate

When a stochastic sequence of projection matrices is available, it can be used to project the population size over time, and the stochastic growth rate can be defined from the resulting population process as the asymptotic time-averaged growth rate:

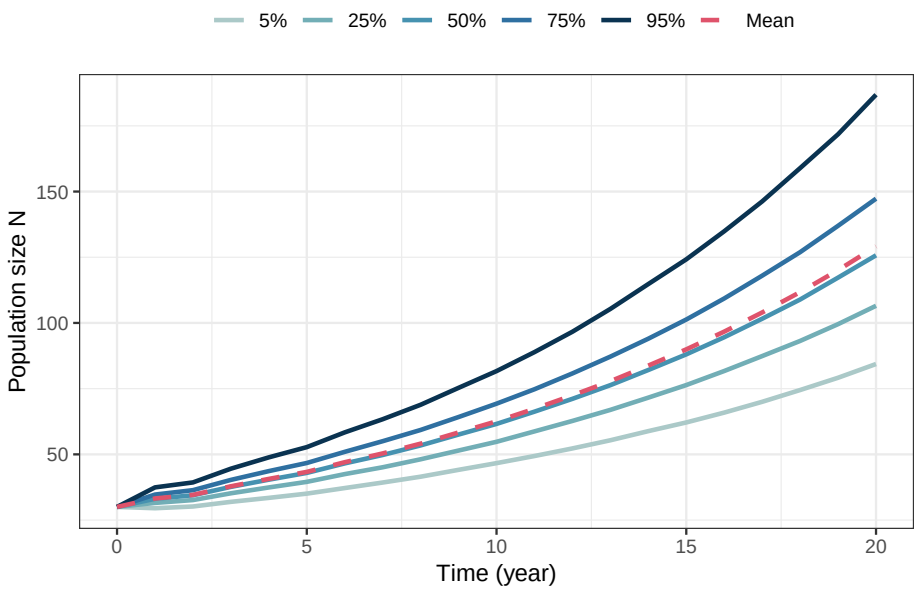


Figure 5.8: Percentiles and mean from 10000 simulated trajectories of the stochastic bird model, with same parameter values as in figure 5.6.

$$r_S = \lim_{t\to\infty} \frac{1}{t}[\ln N_t - \ln N_0], \tag{5.10}$$

This value can easily estimated from simulations, once the sequence of stochastic matrices is determined.

The following code can be used to calculate the stochastic growth rate following the definition of equation (5.10), given a sequence of projection matrices. The cutoff argument removes the first time steps (corresponding to cutoff number), where transient fluctuations can be large unless the population starts at the stable distribution.

```
EstimateSGR <- function(Amats, n0, cutoff=20){
  Tmax <- dim(Amats)[3]
    if(cutoff>=Tmax){
      warning("cutoff must be lower
              than the number of
              matrices in Amats")
      }
  project <- projection.stochastic(Amats=Amats, n0=n0)
  Ntot <- apply(project[,-1],1,sum)
  logN <- log(Ntot[cutoff:length(Ntot)])
  TM <- length(logN)
```

```
  (logN[TM]-logN[1])/TM
}
```

Note that the estimate becomes more accurate the longer the sequence, and the closer the initial population vector is to the stable distribution from the constant (mean) environment (or the higher the cutoff). If the growth rate is very large or small, numerical issues can arise affecting the estimation.

This estimate is based on only one run of the process, and the accuracy can be improved by running it many times.

Bird example

Applying the function in the bird example (for a longer sequence of 1000 projection matrices), we get the following estimate of the stochastic growth rate:

```
set.seed(25)
#Generate sequence of stochastic matrices:
Amats.bird2 <- Amat.array(MatA=Abird.pre,
                          MatU=Ubird.pre,
                          MatF=Fbird.pre,
                          tmax=1000,
                          VarF=rep(0.03, 3),
                          VarS=rep(0.03, 3))

#Estimate growth rate
rs.bird <- EstimateSGR(Amats = Amats.bird2,
                       n0=rep(10,3))

round(rs.bird, 4)
```

```
## [1] 0.0704
```

This means that $r_s \approx 0.0704$ for the bird example. In the constant environment we had $r = \ln \lambda \approx 0.068$ (you can check that you get approximately this value by setting the variances to zero in the above function).

5.4.3 Small noise approximation

So far we have relied on simulations of the population process to estimate the stochastic growth rate, which does not provide a good understanding of how it depends on the variance and covariance of the underlying matrix elements.

Assuming small environmental fluctuations (i.e. that no projection matrix would be very different from the mean matrix), Tuljapurkar [1982] derived the follow-

ing useful approximation for the stochastic growth rate that links it to the underlying matrix elements:

$$r_S \approx r - \frac{1}{2\lambda^2} \sum_i \sum_j \sum_k \sum_l \frac{\partial \lambda}{\partial a_{ij}} \frac{\partial \lambda}{\partial a_{kl}} \mathrm{Cov}(A_{ij}, A_{kl}) \tag{5.11}$$

$$= r - \frac{1}{2\lambda^2} \sigma_e^2. \tag{5.12}$$

We see that equation (5.12) is similar to the approximation of the Lewontin-Cohen model (equation (5.6)), which is not strange since they are both based on a Taylor approximation (mathemtacal details are outside the scope of this compendium). THe small noise approximation is useful because it shows how the stochastic growth rate is related to the sensitivities of the mean projection matrix, and the variances (and covariances) between matrix elements.

The small noise approximation is relevant to the topic of **demographic buffering**. Demographic buffering, or simply 'buffering', refers to the common observation that vital rates to which λ (in the mean environment) shows a high sensitivity in the mean environment often show low temporal variability. This is often seen as a strategy to mitigate negative impacts of variability, which would be most severe if it affected these vital rates that are so important to the growth in the average environment. However, because the variance of vital rates is mathematically constrained by the mean (e.g. for survival, a high mean automatically implies that the variance must be low) it is difficult to tease appart the underlying mechanisms behind an observed low variance and high mean for any given vital rate. In contrast to bet-hedging, buffering refers to any case of low variation in vital rates, without any assumption about any cost to the mean.

5.5 Exercises

Exercise 5.1

Start with the flowering plant from exercise 2.2 and exercise 3.2. Assume a pre-reproductive census, and use the following projection matrix for the mean environment (increased fecundity to get $\lambda > 1$):

```
x <- 0:4
nx <- c(12376, 4233, 1790, 340, 268)
mx <- c(0, 0, 0, 2.1, 3.2)
lx <- nx/nx[1]
px <- c(lx[2:5]/lx[1:4],0)
```

```
C <- 8

APlant <- Create.Amat(Svec=px[2:5] , Fvec=px[1]*mx[2:5]*C)
FPlant<- UPlant<- APlant

FPlant[-1,] <- 0
UPlant[1,] <- 0

round(APlant,3)
```

```
##       [,1] [,2]  [,3]  [,4]
## [1,] 0.000 0.00 5.746 8.756
## [2,] 0.423 0.00 0.000 0.000
## [3,] 0.000 0.19 0.000 0.000
## [4,] 0.000 0.00 0.788 0.000
```

1. Add stochasticity to the fertility coefficients using the same approach as for the bird example in this chapter, and make some plots of the projected growth of the age classes for different values of the variances. Interpret what happens.
2. Add stochasticity to the survival coefficients instead (but not in fertility) and make some plots of the projected growth as in the previous point. Do you notice a difference?
3. Think about possible bet hedging strategies the plant could use to optimize fitness in a varying environment. Do you see any signs of bet-hedging in its current life history?

Exercise 5.2

Consider the black-headed gull example from chapter 4.4.2, with two different habitats. Instead of the migration pattern described in the example (where offspring migrate), assume that migration is random between the two environments, so that each year each gull ends up in either habitat by chance. The two projection matrices describing each habitat are given by

$$\mathbf{A}_{\text{good}} = \begin{bmatrix} 0 & 0.096 & 0.16 & 0.224 & 0.32 \\ 0.8 & 0 & 0 & 0 & 0 \\ 0 & 0.82 & 0 & 0 & 0 \\ 0 & 0 & 0.82 & 0 & 0 \\ 0 & 0 & 0 & 0.82 & 0.82 \end{bmatrix},$$

and

$$\mathbf{A}_{\text{poor}} = \begin{bmatrix} 0 & 0.1 & 0.16 & 0.2 & 0.2 \\ 0.8 & 0 & 0 & 0 & 0 \\ 0 & 0.82 & 0 & 0 & 0 \\ 0 & 0 & 0.82 & 0 & 0 \\ 0 & 0 & 0 & 0.82 & 0.82 \end{bmatrix}.$$

1. What is the value of λ in each habitat?
2. Write an R function that returns a series of gull projection matrices for a given number of time steps (returned as an array where the third dimension is time), assuming a random draw each year (hint: the function `sample()` can be used to make random draws. For instance, `sample(c("A", "B"), 10, replace=TRUE)` will return a vector of 10 values of either "A" or "B".).
3. Calculate the mean growth rate and the stochastic growth rate for the black-headed gull assuming such random migration between the habitats. Make a quantile plot of population growth and highlight the growth trajectories corresponding to growth of the median and mean population.
4. In the `sample()` function there is an argument `prob` that can be used to set different probabilities for each environment. For instance, with `sample(c("A", "B"), 100, replace=TRUE,prob=c(.8, .2))` there is a probability of 0.8 of drawing an "A" and 0.2 of drawing a "B" for each sampling event, resulting in many more "A"'s over time. Modify the previous code and consider what happens to the population growth if the probability of ending up in the good environment is higher, or lower.

Bibliography

Appendix A

Suggested solutions to exercises

A.1 Chapter 1 exercises

Exercise 1.1

1. Make R function for exponential growth:

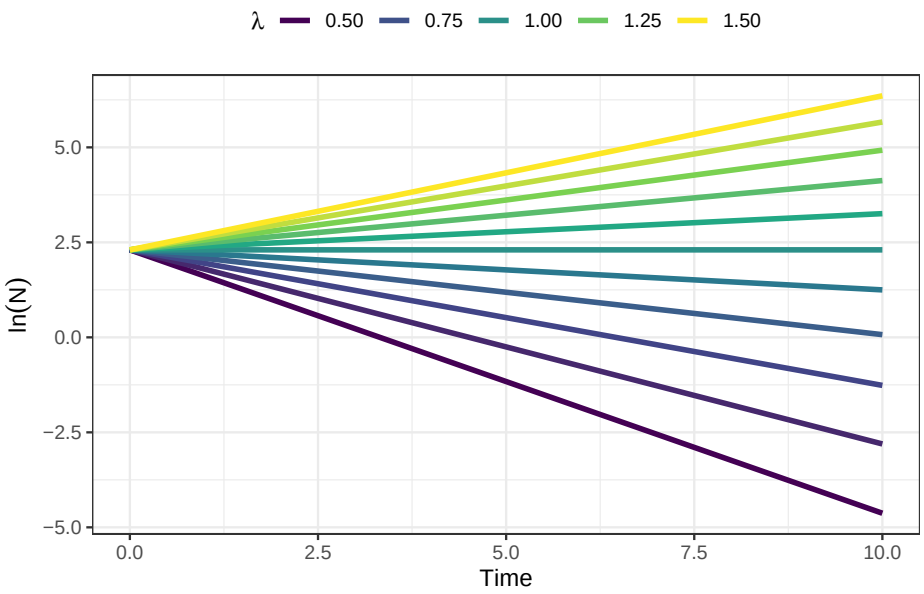
```
#lambda: Multiplicative growth rate
#N0: Initial population size
#Tmax: Number of time steps

Exp.growth <- function(lambda, N0, Tmax){
        N0*lambda^(0:Tmax)
}

#test:
Exp.growth(lambda=1.3, N0=10, Tmax=5)
```

```
## [1] 10.0000 13.0000 16.9000 21.9700 28.5610 37.1293
```

2. Plot population growth for different values of λ:

```
#Make a vector of lambda values:
lambdavec <- seq(.5, 1.5, by=.1)

#Store growth results for each lambda value in a data frame:
results <- expand.grid(lambda=lambdavec, Time=0:10)
for(i in 1:length(lambdavec)){
```

```
        results$Popsize[results$lambda==lambdavec[i]] <-
          Exp.growth(lambda=lambdavec[i],
                N0=10,
                Tmax=10)
}

#Store log population size (better for plotting)
results$LogSize <- log(results$Popsize)

#Plot with ggplot:
ggplot(results) +
    geom_line(aes(x=Time, y=LogSize, col=lambda,
                  group=lambda ), lwd=1.2) +
    theme_bw() +
    scale_color_viridis_c()+
    theme(legend.position = "top" )+
    labs( x="Time", y=expression(ln(N)),
          color=expression(lambda))+
    guides(color = guide_legend(nrow = 1,
                                byrow = TRUE))
```



Exercise 1.2

1.

Value of λ: If the population grows by 1% per day, that means $\lambda \approx 1.01$, with

a unit of days.

2.

Doubling time: We denote the doubling time as t_2. Inserting $N = 2N_0$ in the equation of exponential growth, we get:

$$\begin{aligned} 2N_0 &= N_0\lambda^{t_2} \\ \lambda^{t_2} &= 2 \\ t_2 \ln\lambda &= 2 \\ t_2 &= \frac{2}{\ln\lambda} = \frac{2}{\ln 1.01} \approx 201. \end{aligned}$$

The population will take approximately 201 days to double in size.

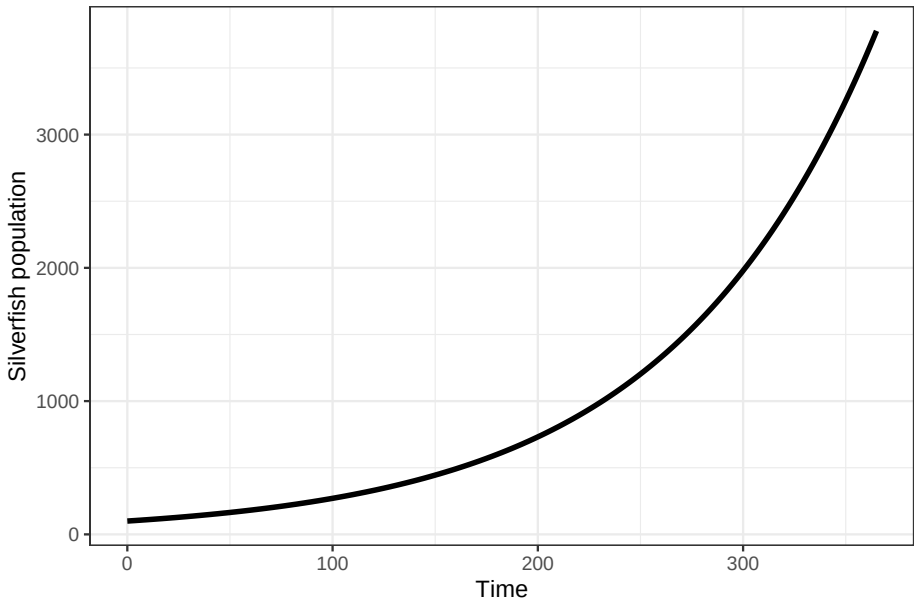
3.

Population size after 1 year: One year is approximately 365 days, so with a starting populatoin of 100 individuals we get

$$N = 100 \cdot 1.01^{365} \approx 3778.$$

Plot population growth over a year:

```
sgrowth <- Exp.growth(lambda=1.01, N0=100, Tmax=365)

#Store as data frame for plotting:
silverfish <- data.frame(sgrowth, Time=0:365)

#Plot with ggplot:
ggplot(silverfish) +
    geom_line(aes(x=Time, y=sgrowth), lwd=1.2) +
    theme_bw() +
    labs( x="Time", y="Silverfish population")
```



A.2 Chapter 2 exercises

Exercise 2.1

1.

The number of children includes boys and girls, so to get R_0 we divide by half (assuming equal sex ratio at birth). Using the approximation ($\lambda \approx \exp\left(\frac{\ln R_0}{T_C}\right)$), we can add values of λ to the table (here assuming a generatio ntime of $T_C = 25$ or $T_C = 30$):

Country	Children per woman	R_0	λ_{25}	λ_{30}
Mali	5.8	2.90	1.044	1.036
Tunisia	2.2	1.10	1.004	1.003
Australia	1.7	0.85	0.994	0.995
Norway	1.6	0.80	0.991	0.993
South Korea	0.9	0.45	0.969	0.974

As generation time T_C increases, λ increases if the population is declining ($\lambda < 1$), but declines if the population is growing (note: this exercise ignores immigration and emigration so the value of λ does not represent the real population trends in these countries.)

2.

To answer this question we use the values λ_{25}, and apply the formula from exercise 1.2 to find the doubling time ($t_2 = 2/\ln \lambda_{25}$):

Country	Children per woman	R_0	λ_{25}	Doubling time
Mali	5.8	2.90	1.044	46.961
Tunisia	2.2	1.10	1.004	524.603
Australia	1.7	0.85	0.994	-307.656
Norway	1.6	0.80	0.991	-224.071
South Korea	0.9	0.45	0.969	-62.617

Declining populations ($\lambda < 1$) get negative doubling rates. Naturally these populations cannot double in size with negative growth. The growing populations ($\lambda > 1$) have a doubling time of 47 years (Mali) and 525 years (Tunisia).

3.

The exponential growth model is too simple to describe long-term growth of human population sizes. It describes current trends of growth given the current rates of survival and fertility. Both survival and fertility change over time, and the annual population growth rate is not constant, but has declined over time in most countries. The generation time also varies much among populations and will change over time (as it is a function of the survival and fertility rates). In particular for countries that currently grow fast we should be careful with making long-term extrapolations based on the exponential growth model. Finally, this exercise ignores immigration and emigration, which can be a large factor affecting population growth.

Exercise 2.2

1.

The age at first reproduction is 3 years. This is the first age where $m_x > 0$ in the life table.

2.

Creating the life table in R (here we use the function ´data.frame()´, other options can also be used, like `tibble()` from tidyverse):

```
x <- 0:4
nx <- c(12376, 4233, 1790, 340, 268)
mx <- c(0, 0, 0, 2.1, 3.2)

PlantTable <- data.frame(x, nx, mx)
```

3.

Adding vectors of cumulative survival probability l_x and annual survival probability p_x to the life table:

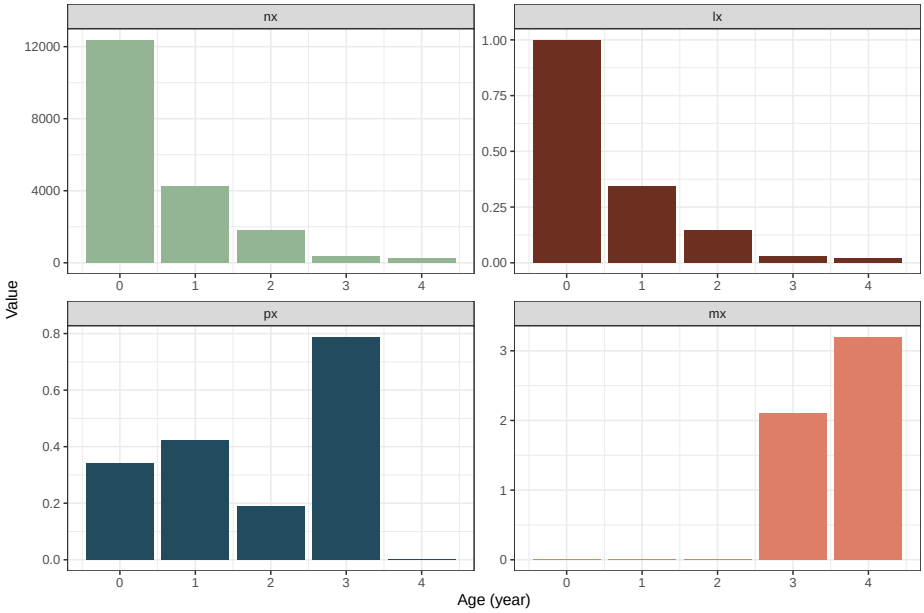
```
lx <- nx/nx[1] #divide all elements of nx by the first.
px <- c(lx[2:5]/lx[1:4],0) #by definition the last value is zero.

PlantTable$lx <- lx
PlantTable$px <- px

PlantTable
```

```
##   x    nx  mx         lx        px
## 1 0 12376 0.0 1.00000000 0.3420330
## 2 1  4233 0.0 0.34203297 0.4228679
## 3 2  1790 0.0 0.14463478 0.1899441
## 4 3   340 2.1 0.02747253 0.7882353
## 5 4   268 3.2 0.02165482 0.0000000
```

4.

Plots of each life table column (n_x, l_x, p_x, m_x) as a function of age x:

```
#Convert to long table format for plotting:
plant.long <- PlantTable %>%
  pivot_longer(c(nx, lx, px,  mx),
               names_to = "VitalRates",
               values_to = "Value")

ggplot(plant.long) +
  geom_col(aes(x=x, y=Value,
               fill=VitalRates), lwd=1.2)+
  scale_fill_manual(values=colors4)+
  theme_bw()+
  facet_wrap(~factor(VitalRates,
                     levels=c("nx", "lx" , "px", "mx")),
             scales="free")+
  theme_bw() +
  labs( x="Age (year)", y="Value")+
  theme(legend.position = "none" )
```



The shape of l_x follows the decline in cohort size over age, as it should. The annual survival probability fluctuates between ages but is quite low in the first years until the plant starts to reproduce at age 3. From age 3 to 4 the survival probability is much higher, almost 0.8, but then no plants survive after age 4 (last reproduction). Since all remaining plants die in the final year, the plants likely invest all resources in reproduction this year, and fecundity is higher at the second reproduction.

5.

Mean lifetime reproduction R_0 from the life table:

```
R0_Plant <- sum(PlantTable$lx*PlantTable$mx)

round(R0_Plant, digits=3)
```

```
## [1] 0.127
```

The value of R_0 is very low. Each plant is replaced by only 0.13 new plants on average. This means that $\lambda < 1$.

6.

Generation time T_C from the life table:

```
TC_Plant <- sum(PlantTable$x*PlantTable$lx*PlantTable$mx)/R0_Plant

round(TC_Plant, digits=3)
```

```
## [1] 3.546
```

This is the cohort generation time, and shows the average age at which the R_0 offspring are produced.

Approximation of λ based on R_0 and T_C:

```
lambda_Plant <- exp(log(R0_Plant)/TC_Plant)
#round(lambda_Plant, digits=3)
```

Thus, the population growth rate is very low at $\lambda \approx 0.559$. Almost half the population is lost each year.

7.

Solving the Euler-Lotka equation for λ:

```
#Rearranging the Euler-Lotka function
#to get an expression equal to zero:
EulerLotka <- function(lambda, x, lx, mx){
  sum(lambda^(-x)*lx*mx)-1
}

#Define search range for lambda in uniroot
lambda.range <- c(.1,2)

#Apply uniroot to find lambda, store the result
lambda_Plant2 <- uniroot(EulerLotka,
                         x=PlantTable$x,
                         lx=PlantTable$lx,
                         mx=PlantTable$mx,
                         interval=lambda.range)$root

#round(lambda_Plant2, digits=3)
```

The value calculated from the Euler-Lotka equation is $\lambda \approx 0.565$, a bit higher than the approximated value.

Exercise 2.3

1.

According to table 2.2, the fecundity values of age 2 and 3 are $m_2 = 3$ and $m_3 = 6$. To get the clutch size these values should be multiplied by 2 (assuming equal sex ratio), and divided by the survival probability. This gives an average clutch size of $M_2 = 3 \cdot 2/0.9 = 6.67$ eggs at age 3, and $M_2 = 6 \cdot 2/0.9 = 13.33$ eggs at age 4.

Thus, older parents produce larger clutches than younger ones, in this (fictive) example.

2.

Under the assumptions of the Lack clutch, the trade-off is defined as $c = \frac{1}{2N_e^*}$. At age 2, the Lack clutch size is 6.67 and we thus get $c = \frac{1}{2 \cdot 6.67} \approx 0.074$. At age 3 the Lack clutch size is 13.33 and the trade-off is $c = \frac{1}{2 \cdot 13.33} \approx 0.038$. Remember that the stronger the trade-off, the lower the clutch size. If the clutch size at each age is determined by this trade-off only (as the Lack clutch assumes), this means that the trade-off is reduced with age for this example. In reality, physical and behavioral constraints also limit clutch size of young versus older birds, in particular in first-time breeders versus experienced breeders.

3.

To solve this we use the Euler-Lotka-equation, and insert the information we have (the value of $\lambda = 1.07$ is calculated in the example for the Euler-Lotka equation):

$$\begin{aligned} \sum_x \lambda^{-x} l_x m_x &= 1 \\ \lambda^{-3} l_3 m_3 &= 1 \\ 1.07^{-3} 0.108 m_3 &= 1 \\ m_3 &= 1.07^3 / 0.108 \approx 11.34. \end{aligned}$$

Thus, the fecundity of the related species is higher, which makes sense since it only reproduces at age 3 while still having the same fitness as the example bird.

4.

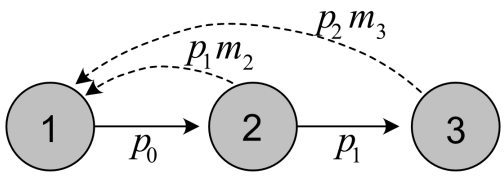
In Cole's paradox, the semelparous species only needs one more offspring per clutch to get the same fitness as the iteroparous species. The result of the previous question tells us that the semelparous bird needs a much larger clutch size to match the fitness of the iteroparous species (or example species), so clearly semelparity is not as 'cheap' here as in Cole's example.

Cole assumed a survival probability of 1 for the iteroparous species, and that both species reproduced from year 1, neither of which apply to the two bird species here. In both species there is a difference between juveniles and adults (juveniles do not reproduce), also not fitting the assumptions in Cole's paradox.

A.3 Chapter 3 exercises

Exercise 3.1

Life cycle for the bird example, with post-reproductive census:



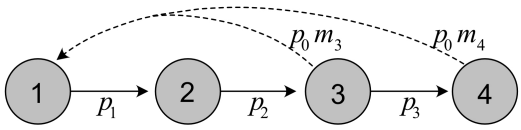
Note that each age class is numbered from 1 to 3, although they correspond to actual age 0 to 2 in this case.

Exercise 3.2

Pre-reproductive census:

1.

Life cycle graph of plant:



2.

Vectors of fertility and survival coefficients:

```
px <- PlantTable$px
mx <- PlantTable$mx

fertility.plant <- px[1]*mx[2:5]
survival.plant <- px[2:5]
```

3.

Define **A**, **U** and **F**:

```
AmatPlant <- Create.Amat(Svec=survival.plant,
                         Fvec=fertility.plant)
FmatPlant <- UmatPlant <- AmatPlant

FmatPlant[-1,] <- 0
UmatPlant[1,] <- 0

AmatPlant
```

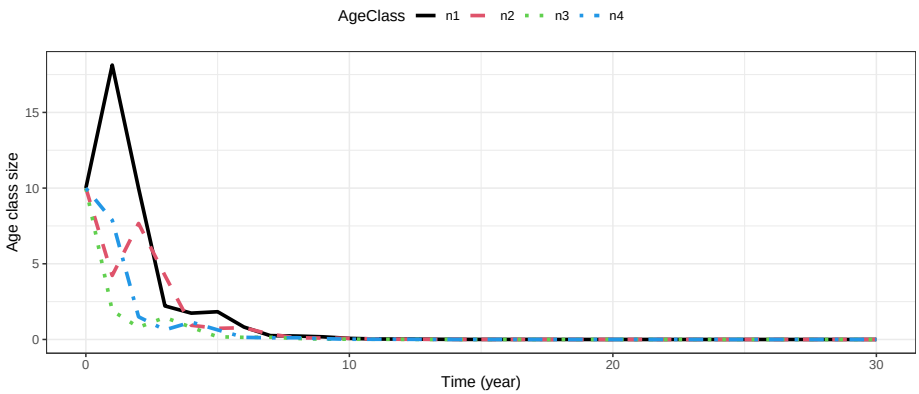
```
##           [,1]      [,2]      [,3]     [,4]
## [1,] 0.0000000 0.0000000 0.7182692 1.094505
```

```
## [2,] 0.4228679 0.0000000 0.0000000 0.000000
## [3,] 0.0000000 0.1899441 0.0000000 0.000000
## [4,] 0.0000000 0.0000000 0.7882353 0.000000
```

4.

Project age classes over 30 time steps:

```
#Return matrix of projected growth of each age class:
Pop.plant <- projection(MatA=AmatPlant,
                        n0=rep(10, 4),
                        Tmax=30)

#Long format for plotting:
Pop.plant.long <- Pop.plant %>%
  pivot_longer(c(-time),
               names_to = "AgeClass",
               values_to = "Value")

#Plot
ggplot(Pop.plant.long) +
  geom_line(aes(x=time,y=Value, col=AgeClass,
                linetype=AgeClass), lwd=1.2)+
  theme_bw() +
  scale_color_manual(values=c(1,2,3,4))+
  scale_linetype_manual(values=c(1,2,3,4))+
  labs( x="Time (year)", y="Age class size")+
  theme(legend.position = "top")
```



5.

Calculate λ, R_0 and G using the matrices:

```
res.plant <- uvlambda(AmatPlant)
lambda.plant <- res.plant$lambda
R0.plant <- R0function(UmatPlant,
```

```
                        FmatPlant)$R0
G.plant <- GenTime(AmatPlant,
                   FmatPlant)

ResVec <- c(lambda.plant,
            R0.plant,
            G.plant)
names(ResVec) <- c("lambda", "R0", "G")
round(ResVec,3)
```

```
## lambda     R0      G
##  0.565  0.127  3.680
```

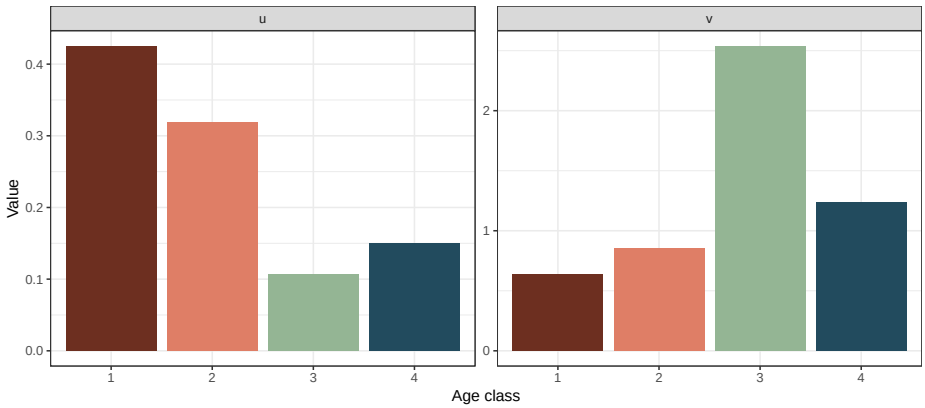
These are approximately the same values as found from the life table. The generation time is slightly different because it is not the same measure. For the life table calculations we used the cohort generation time, while for the matrix model we used the mean age of mothers at the stable distribution.

6.

Calculate and plot **u** and **v**:

These values were calculated in the previous point using the R function `uvlambda()`. To plot them:

```
#Create data frame for plotting
plantframe <- data.frame("AgeClass"=factor(1:4),
                         "u"=res.plant$u,
                         "v"=t(res.plant$v))

#Long format for plotting:
plantframe.long <- plantframe %>%
  pivot_longer(c(u,v),
               names_to = "Vector",
               values_to = "Value")


#Plot
ggplot(plantframe.long) +
  geom_col(aes(x=AgeClass, y=Value,
               fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(Vector),ncol=2,
             scales="free")+
  scale_fill_manual(values=colors4)+
  labs( x="Age class", y="Value")+
  theme(legend.position = "none" )
```



The stable age structure tells us the proportion of individuals in each age class when the population has reached the stable growth phase. In this case the stable age structure shows a higher proportion of individuals in age class 4 than in age class 3. This is uncommon for age structured populations (usually the stable age structure declines with age), and is due to the extremely low value of λ.

The reproductive values tell us how much individuals in each age class will contribute to the future population, relative to other age classes (when the population has reached the stable growth phase). In this case the reproductive value is highest in age class 3, followed by age class 4, the two reproductive classes.

7.

Increase fecundity by factor c to get $\lambda > 1$: Based on trial and error (apply the code several times for different values of c) we find the value of c that increases the fecundity enough to get lambda >1:

```
C <- 8 #Updated through trial and error
AmatPlantNew <- Create.Amat(Svec=px[2:5],
                            Fvec=px[1]*mx[2:5]*C)
FmatPlantNew <- UmatPlantNew <- AmatPlantNew

FmatPlantNew[-1,] <- 0
UmatPlantNew[1,] <- 0

uvlambda(AmatPlantNew)$lambda
```

```
## [1] 1.00446
```

With a multiplying factor of 8 (increasing fecundity 8 times) we can get λ over 1. Calculate the other parameters and return in vector:

```
res.plantNew <- uvlambda(AmatPlantNew)
lambda.plantNew <- res.plantNew$lambda
R0.plantNew <- R0function(UmatPlantNew,
```

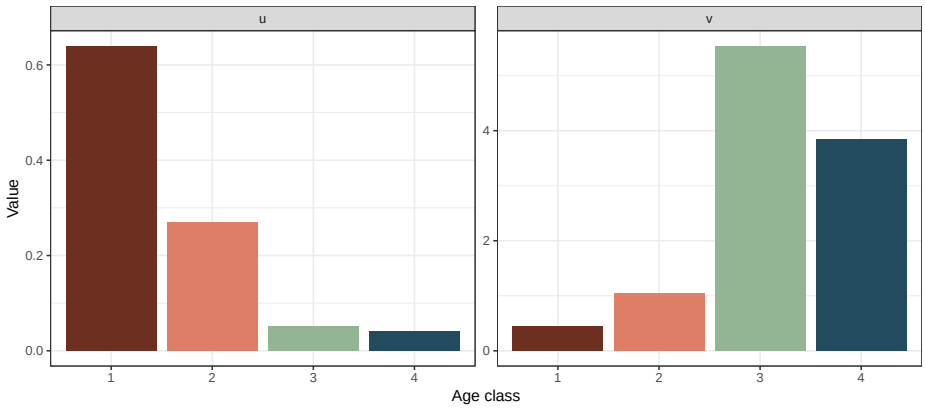
```
                            FmatPlantNew)$R0
G.plantNew <- GenTime(AmatPlantNew,
                      FmatPlantNew)

ResVecNew <- c(lambda.plantNew,
               R0.plantNew,
               G.plantNew)
names(ResVecNew) <- c("lambda", "R0", "G")
round(ResVecNew,3)
```

```
## lambda     R0      G
##  1.004  1.016  3.545
```

Compared to before, R_0 has increased a lot (it is now >1 which it has to be because $\lambda > 1$), while the generation time has only moderately changed, showing a slight decrease (thus the average age of mothers has decreased slightly).

Plot the new stable structure and reproductive values:

```
#Create data frame for plotting
plantframeNew <- data.frame("AgeClass"=factor(1:4),
                            "u"=res.plantNew$u,
                            "v"=t(res.plantNew$v))

#Long format for plotting:
plantframeNew.long <- plantframeNew %>%
  pivot_longer(c(u,v),
               names_to = "Vector",
               values_to = "Value")


#Plot
ggplot(plantframeNew.long) +
  geom_col(aes(x=AgeClass, y=Value, fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(Vector),ncol=2,scales="free")+
  scale_fill_manual(values=colors4)+
  labs( x="Age class", y="Value")+
  theme(legend.position = "none" )
```



Now that $\lambda > 1$, the stable age structure is declining over age. Compared to before it is also more skewed towards the younger age classes. The reproductive value is still highest in age class 3, but the difference is now larger between the two reproducing classes and the young non-reproducing classes.

8.

Increase survival by factor c to get $\lambda > 1$:

Based on trial and error (apply the code several times for different values of c) we find the value of c that increases the survival enough to get lambda >1 (NB: Note that we need to update both the fertility and survival coefficients):

```
C <- 1.8 #Updated through trial and error
AmatPlantNew2 <- Create.Amat(Svec=px[2:5]*C,
                             Fvec=px[1]*mx[2:5]*C)
FmatPlantNew2 <- UmatPlantNew2 <- AmatPlantNew2

FmatPlantNew2[-1,] <- 0
UmatPlantNew2[1,] <- 0

uvlambda(AmatPlantNew2)$lambda
```

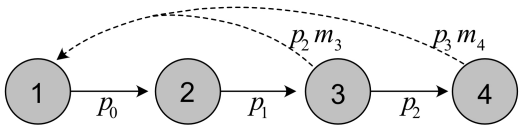
```
## [1] 1.016965
```

With a multiplying factor of 1.8 for the survival vector to get λ over 1, however this means that the survival probability of the third age class is > 1, which is not realistic/possible.

Post-reproductive census:

1.

Life cycle graph of the plant:



2.

Vectors of fertility and survival coefficients:

```
#corresponding to age class 1:4, age 0:3:
fertility.plant.post <- px[1:4]*mx[2:5]
survival.plant.post <- c(px[1:3],0)
```

3.

Define **A**, **U** and **F**:

```
AmatPlant.post <- Create.Amat(Svec=survival.plant.post,
                              Fvec=fertility.plant.post)
FmatPlant.post <-
  UmatPlant.post <- AmatPlant.post

FmatPlant.post[-1,] <- 0
UmatPlant.post[1,] <- 0

AmatPlant.post
```

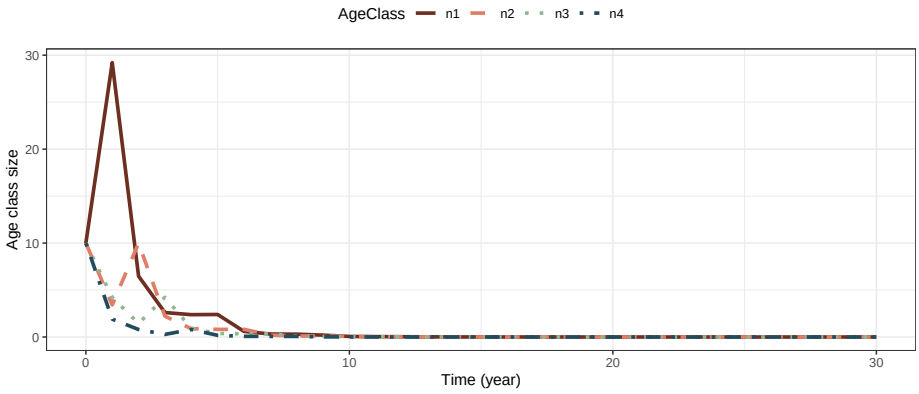
```
##          [,1]      [,2]      [,3]     [,4]
## [1,] 0.000000 0.0000000 0.3988827 2.522353
## [2,] 0.342033 0.0000000 0.0000000 0.000000
## [3,] 0.000000 0.4228679 0.0000000 0.000000
## [4,] 0.000000 0.0000000 0.1899441 0.000000
```

4.

Project age classes over 30 time steps:

```
#Return matrix of projected growth of each age class:
Pop.plant.post <- projection(MatA=AmatPlant.post,
                             n0=rep(10, 4),
                             Tmax=30)

#Long format for plotting:
Pop.plant.long.post <- Pop.plant.post %>%
  pivot_longer(c(-time),
               names_to = "AgeClass",
               values_to = "Value")

#Plot
```

```
ggplot(Pop.plant.long.post) +
  geom_line(aes(x=time,y=Value, col=AgeClass,
                linetype=AgeClass), lwd=1.2)+
  theme_bw() +
  scale_color_manual(values=colors4)+
  scale_linetype_manual(values=1:4)+
  labs( x="Time (year)", y="Age class size")+
  theme(legend.position = "top" )
```



5.

Calculate λ, R_0 and G using the matrices:

```
res.plant.post <- uvlambda(AmatPlant.post)
lambda.plant.post <- res.plant.post$lambda
R0.plant.post <- R0function(UmatPlant.post,
                            FmatPlant.post)$R0
G.plant.post <- GenTime(AmatPlant.post,
                        FmatPlant.post)

ResVec.post <- c(lambda.plant.post,
                 R0.plant.post,
                 G.plant.post) #Store results
names(ResVec.post) <- c("lambda", "R0", "G")
round(ResVec.post,3)
```

```
## lambda     R0      G
##  0.565  0.127  3.680
```

The values are the same as with the pre-reproductive census model.

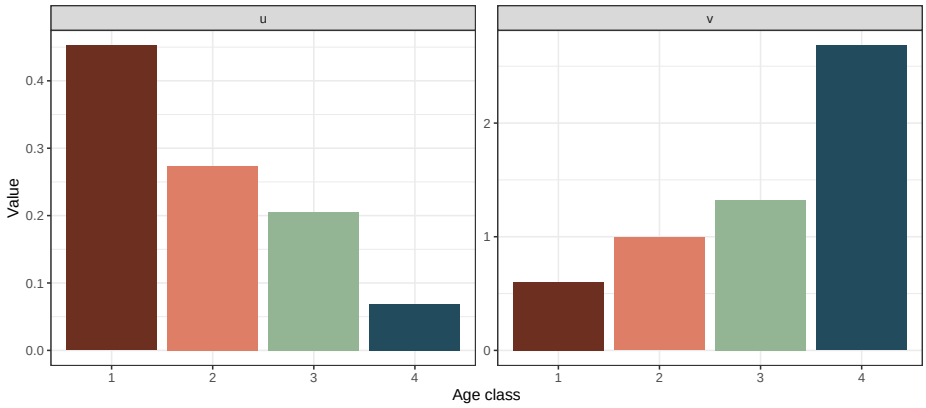
6.

Calculate and plot **u** and **v**:

These values were calculated in the previous point using the R function

uvlambda(). To plot them:

```
#Create data frame for plotting
plantframe.post <- data.frame("AgeClass"=factor(1:4),
                              "u"=res.plant.post$u,
                              "v"=t(res.plant.post$v))

#Long format for plotting:
plantframe.post.long <- plantframe.post %>%
  pivot_longer(c(u,v),
               names_to = "Vector",
               values_to = "Value")


#Plot
ggplot(plantframe.post.long) +
  geom_col(aes(x=AgeClass, y=Value,
               fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(Vector),ncol=2,
             scales="free")+
  scale_fill_manual(values=colors4)+
  labs( x="Age class", y="Value")+
  theme(legend.position = "none" )
```



Now the age classes correspond to actual age 0 to 3, in contrast to the pre-reproductive model where they correspond to age 1 to 4. The stable age structure and reproductive values look different here, since the classes represent other ages. Here the highest reproductive value is found for age class 4, corresponding to age 3. This is in line with the result from the pre-reproductive model where the highest reproductive value was found in age class 3, there corresponding to age 3.

7.

Increase fecundity by factor c to get $\lambda > 1$:

Based on trial and error (apply the code several times for different values of c) we find the value of c that increases the fecundity enough to get lambda >1:

```
#Updated through trial and error :
C <- 8
AmatPlantNew.post <- Create.Amat(
  Svec= c(px[1:3],0),
  Fvec=px[1:4]*mx[2:5]*C)

FmatPlantNew.post <-
  UmatPlantNew.post <- AmatPlantNew.post

FmatPlantNew.post[-1,] <- 0
UmatPlantNew.post[1,] <- 0

uvlambda(AmatPlantNew.post)$lambda
```

```
## [1] 1.00446
```

We need the same factor of 8 (increasing fecundity 8 times) to get λ over 1.

The other parameters

```
res.plantNew.post <- uvlambda(AmatPlantNew.post)
lambda.plantNew.post <- res.plantNew.post$lambda
R0.plantNew.post <- R0function(UmatPlantNew.post,
                               FmatPlantNew.post)$R0
G.plantNew.post <- GenTime(AmatPlantNew.post,
                           FmatPlantNew.post)

ResVecNew.post <- c(lambda.plantNew.post,
                    R0.plantNew.post,
                    G.plantNew.post)
names(ResVecNew.post) <- c("lambda", "R0", "G")
round(ResVecNew.post,3)
```

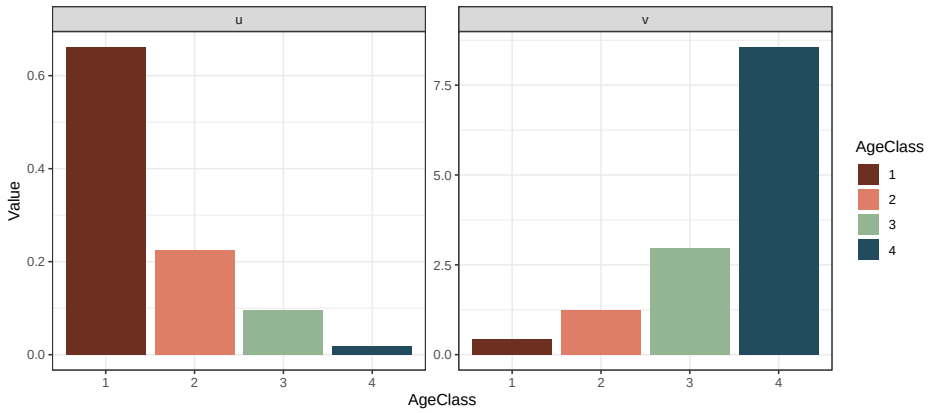
```
## lambda     R0      G
##  1.004  1.016  3.545
```

These values are the same as in the pre-reproductive (updated) model.

Plot new stable structure and reproductive values:

```
#Create data frame for plotting
plantframeNew.post <- data.frame("AgeClass"=factor(1:4),
             "u"=res.plantNew.post$u,
             "v"=t(res.plantNew.post$v))
```

```
#Long format for plotting:
plantframeNew.post.long <- plantframeNew.post %>%
  pivot_longer(c(u,v),
               names_to = "Vector",
               values_to = "Value")

#Plot:
ggplot(plantframeNew.post.long) +
  geom_col(aes(x=AgeClass, y=Value,
               fill=AgeClass ), lwd=1.2)+
  theme_bw() +
  facet_wrap(vars(Vector),ncol=2,
             scales="free")+
  scale_fill_manual(values=colors4)
```




```
  labs( x="Age class", y="Value")
```

```
## $x
## [1] "Age class"
##
## $y
## [1] "Value"
##
## attr(,"class")
## [1] "labels"
```

```
  theme(legend.position = "none")
```

```
## List of 1
##  $ legend.position: chr "none"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi FALSE
##  - attr(*, "validate")= logi TRUE
```

Compared to before, the stable structure is more skewed towards the younger age classes. The reproductive value is still highest in age class 4, but the difference is now larger between the classes.

8.

Increase survival by factor c to get $\lambda > 1$:

Based on trial and error (apply the code several times for different values of c) we find the value of c that increases the fecundity enough to get lambda >1:

```
#Updated through trial and error :
C <- 1.8
AmatPlantNew2.post <- Create.Amat(
  Svec= c(px[1:3],0)*C,
  Fvec=px[1:4]*mx[2:5]*C)

FmatPlantNew2.post <-
  UmatPlantNew2.post <- AmatPlantNew2.post

FmatPlantNew2.post[-1,] <- 0
UmatPlantNew2.post[1,] <- 0

uvlambda(AmatPlantNew2.post)$lambda
```

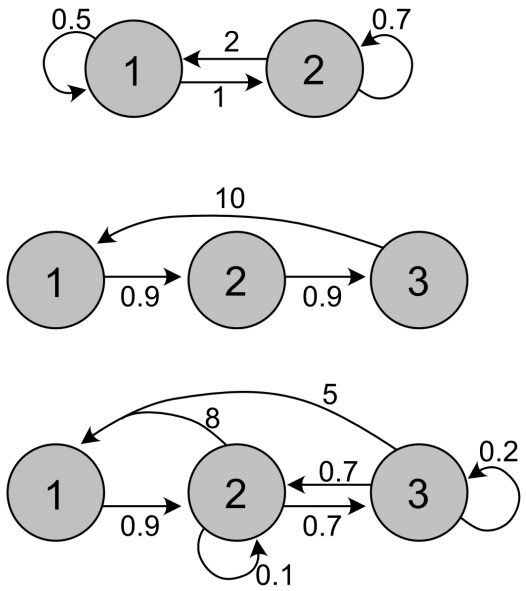
```
## [1] 1.016965
```

We still need a multiplying factor of 1.8 for survival to get λ over 1, and we get the same problem with $p_3 > 1$.

A.4 Chapter 4 exercises

Exercise 4.1

1.



2.

To separate out the vectors of survival probabilities and fertility coefficients, we need the matrices **U** and **F**, instead of only **A**. If we assume that the first row of **A** represent fertilities and the others represent survival and transitions, we can decompose **A** (note that in general this is not an assumption we can make):

```
# Create the projection matrices in R:
Amat1 <- cbind(c(.5, 1),
               c(2,.7))
Amat2 <- cbind(c(0, .9, 0),
               c(0, 0, .9),
               c(10, 0, 0))
Amat3 <- cbind(c(0, .9, 0),
               c(8, .1, .7),
               c(5, .7, .2))

# Create matrices U and F for each example,
#under the assumption mentioned:
Umat1 <- cbind(c(0, 1),
               c(0,.7))
Umat2 <- cbind(c(0, .9, 0),
               c(0, 0, .9),
               c(0, 0, 0))
Umat3 <- cbind(c(0, .9, 0),
               c(0, .1, .7),
               c(0, .7, .2))
```

```
Fmat1 <- cbind(c(.5, 0),
               c(2, 0))
Fmat2 <- cbind(c(0, 0, 0),
               c(0, 0, 0),
               c(10, 0, 0))
Fmat3 <- cbind(c(0, 0, 0),
               c(8, 0, 0),
               c(5, 0, 0))

#Apply the decompose functions to decompose
#each matrix (shown here for the first only)

DecomposeU(Umat1)
```

```
## $Gmat
##      [,1] [,2]
## [1,]    0    0
## [2,]    1    1
##
## $Survival
## [1] 1.0 0.7
```

```
DecomposeF(Fmat1)
```

```
## $Qmat
##      [,1] [,2]
## [1,]    1    1
## [2,]    0    0
##
## $Fertility
## [1] 0.5 2.0
```

3.

The first matrix has only two stages, and the numbers suggest reproduction occurs in both, as well as transition of individuals. This could be a model for two habitats with migration between them.

The middle matrix is a typical Leslie matrix. Note that only the last age class reproduces, so this is a semelparous life history.

The third matrix has many transitions, but based on the numbers the first stage seems to be an offspring stage. The two other stages both reproduce, and there are transitions between them. They could represent two habitats, or two size classes (where both growing and shrinking is possible).

Exercise 4.2

1.

Define the fertility matrix **F**, the survival matrix **U**, and the projection matrix **A**:

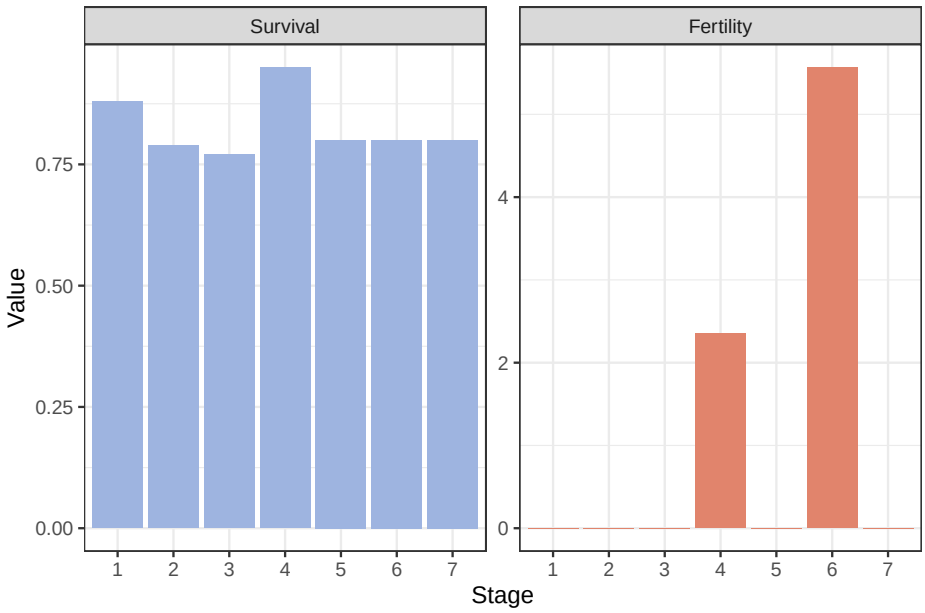
```
#Define vectors
Pvec<- c(0.158, 0.545, 0.539, 0.162, 0.008, 0.136, 0.8)
Gvec <- c(0.722,0.244, 0.231, 0.788 , 0.792, 0.664,0)
Fvec <- c(0, 0, 0, 2.355, 0, 5.568, 0)

#Create matrices:
Umat.isopod <- Fmat.isopod <- matrix(0,7,7)
Umat.isopod[row(Umat.isopod) == col(Umat.isopod) + 1] <- Gvec[1:6]
Umat.isopod[row(Umat.isopod) == col(Umat.isopod)] <- Pvec

Fmat.isopod[1,] <- Fvec
Amat.isopod <- Umat.isopod + Fmat.isopod
```

2.

Decompose the matrices **U** and **F**, plot the stage-specific fertility and survival probability:

```
Svec.isopod <- DecomposeU(Umat.isopod)$Survival
Fvec.isopod <- DecomposeF(Fmat.isopod)$Fertility

#Make a data frame for plotting:
frame.isopod <- data.frame("Stage"=as.factor(1:7),
                           "Survival"=Svec.isopod,
                           "Fertility"=Fvec.isopod)

#Long format:
frame.isopod.long <- frame.isopod %>%
  pivot_longer(c(Survival, Fertility),
               names_to = "VitalRates",
               values_to = "Value")


#Plot with ggplot:
ggplot(frame.isopod.long) +
  geom_col(aes(x=Stage, y=Value, fill=VitalRates ))+
  scale_fill_manual(values=colors2)+
  facet_wrap(~factor(VitalRates,
                     levels=c("Survival", "Fertility")),
             scales="free")+
  theme_bw() +
```

```
  labs( x="Stage", y="Value")+
  theme(legend.position = "none" )
```

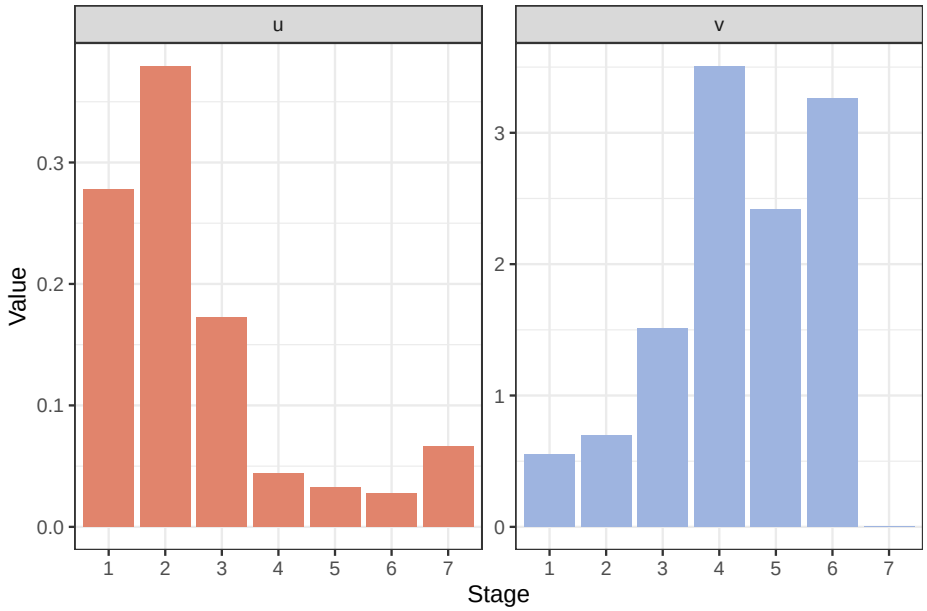


3.

Calculate λ, the stable stage structure and reproductive values, and plot the result:

```
res.isopod <- uvlambda(Amat.isopod)

lambda.isopod <- res.isopod$lam

#Make a data frame for plotting:
resframe.isopod <- data.frame("Stage"=as.factor(1:7),
                              "u"=res.isopod$u,
                              "v"=as.vector(res.isopod$v))

#Long format:
resframe.isopod.long <- resframe.isopod %>%
  pivot_longer(c(u, v), names_to = "uv",
               values_to = "Value")


#Plot with ggplot:
ggplot(resframe.isopod.long) +
  geom_col(aes(x=Stage, y=Value, fill=uv ))+
  scale_fill_manual(values=colors2)+
```

```
  facet_wrap(~factor(uv, levels=c("u", "v")),
             scales="free")+
  theme_bw() +
  labs( x="Stage", y="Value")+
  theme(legend.position = "none" )
```



The growth rate is $\lambda \approx 1.0745$ per month. Per year the growth rate is $\lambda^12 \approx 2.3683$. Thus, the annual growth rate is high, more than doubling the population each year. However, this matrix model assumes that the population can reproduce every month, and reaches a stable structure with individuals in the reproductive stages each month, while in reality there are only two main breeding periods per year [Kammenga et al., 2001]. Thus, while the model can describe the life of one cohort reasonably well (e.g. we can use the projection matrix to project the growth over a year or two starting with a cohort in stage 1), it likely overestimates the population growth rate.

Note that the reproductive value is zero in the final stage, because this is a post-reproductive stage. It does not contribute to population growth, but can still be relevant to consider in the model.

4.

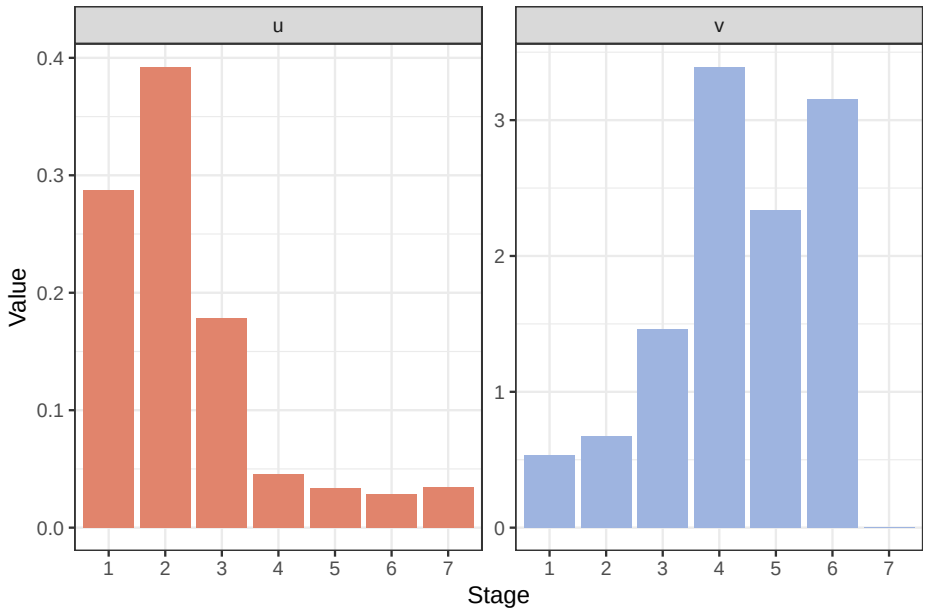
Write function that returns the projection matrix as a function of the input vectors:

```
A.isopod <- function(PV=Pvec, GV=Gvec, FV=Fvec){
  Umat.isopod <- Fmat.isopod <- matrix(0,7,7)
  Umat.isopod[row(Umat.isopod) == col(Umat.isopod) + 1] <- GV[1:6]
```

```
  Umat.isopod[row(Umat.isopod) == col(Umat.isopod)] <- PV
  Fmat.isopod[1,] <- FV
  Amat.isopod <- Umat.isopod + Fmat.isopod
  Amat.isopod
}
```

What happens to λ, $\mathbf{u}$ and $\mathbf{v}$ if you modify the element G_6 (here the element is multiplied by 0.5):

```
#Modify element 6 of the G-vector, e.g. multiply it by 0.5
Amat.isopod2 <- A.isopod(GV=c(0.722,
                              0.244,
                              0.231,
                              0.788,
                              0.792,
                              0.5*0.664,
                              0))

res.isopod2 <- uvlambda(Amat.isopod2)

lambda.isopod2 <- res.isopod2$lam

#Make a data frame for plotting:
resframe.isopod2 <- data.frame("Stage"=as.factor(1:7),
                               "u"=res.isopod2$u,
                               "v"=as.vector(res.isopod2$v))

#Long format:
resframe.isopod.long2 <- resframe.isopod2 %>%
  pivot_longer(c(u, v), names_to = "uv",
               values_to = "Value")

#Plot with ggplot:
ggplot(resframe.isopod.long2) +
  geom_col(aes(x=Stage, y=Value, fill=uv ))+
  scale_fill_manual(values=colors2)+
  facet_wrap(~factor(uv, levels=c("u", "v")),
             scales="free")+
  theme_bw() +
  labs( x="Stage", y="Value")+
  theme(legend.position = "none" )
```



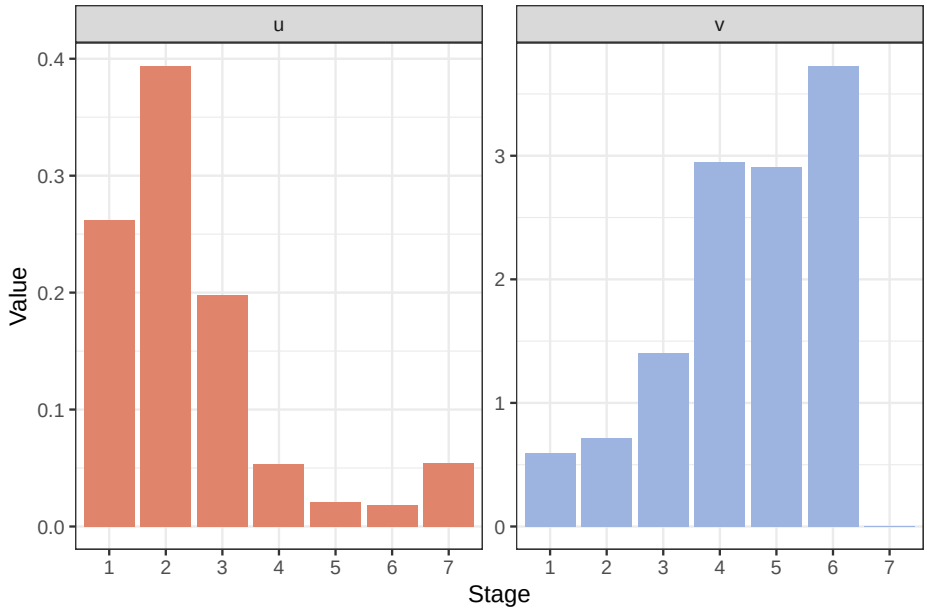
If we use a new value of $G_6 = 0.5 \cdot 0.664$ (half the original), we get $\lambda \approx 1.0745$, exactly the same as before. This is because only the final stage is affected, which is a post-reproductive stage not contributing to population growth. The stable structure changes slightly because we get relatively fewer individuals in the final stage. As a result the reproductive values are also rescaled but otherwise not changed.

5.

Modify the element G_4 instead:

```
#Modify element 6 of the G-vector, e.g. multiply it by 0.5
Amat.isopod3 <- A.isopod(GV=c(0.722,
                              0.244,
                              0.231,
                              0.5*0.788 ,
                              0.792,
                              0.664,
                              0))

res.isopod3 <- uvlambda(Amat.isopod3)

lambda.isopod3 <- res.isopod3$lam

#Make a data frame for plotting:
resframe.isopod3 <- data.frame("Stage"=as.factor(1:7),
                               "u"=res.isopod3$u,
```

```
                                "v"=as.vector(res.isopod3$v))

#Long format:
resframe.isopod.long3 <- resframe.isopod3 %>%
  pivot_longer(c(u, v),
               names_to = "uv",
               values_to = "Value")


#Plot with ggplot:
ggplot(resframe.isopod.long3) +
  geom_col(aes(x=Stage, y=Value, fill=uv ))+
  scale_fill_manual(values=colors2)+
  facet_wrap(~factor(uv, levels=c("u", "v")),
             scales="free")+
  theme_bw() +
  labs( x="Stage", y="Value")+
  theme(legend.position = "none" )
```



If we use a new value of $G_4 = 0.5 \cdot 0.788$ (half the original), we get $\lambda \approx 1.0246$, lower than before. This time the second reproductive stage is also affected, as fewer individuals transition out from stage 4. The highest reproductive value shifts to stage 6 instead of 4, but there are fewer individuals reaching this stage than before.

Exercise 4.3

1.

To define the new matrix $\mathbf{A}_C$ we can use the function defined in exercise 4.2 returning the projection matrix as a function of the vectors P, G, and F:

```
PvecC <- c(0.158, 0.507, 0.501, 0.15, 0.007, 0.126, 0.744)
GvecC <- c(0.722, 0.228, 0.215, 0.733, 0.737, 0.618,0)
FvecC <- c(0, 0, 0, 2.355, 0,0,0)

Amat.isopod.C <- A.isopod(PV=PvecC, GV=GvecC, FV=FvecC)
```

2.

Calculate λ for the new matrix:

```
res.isopod.C <- uvlambda(Amat.isopod.C)
lambda.isopod.C <- res.isopod.C$lam
```

The value of λ in the 'cadmium environment' is $\lambda \approx 0.8941$, much lower than the reference values of $\lambda_R \approx 1.0745$ (exercise 4.2). The difference in λ is $\lambda_C - \lambda_R \approx round(lambda.isopod.C - lambda.isopod, 4)$.

3.

Calculate halfway and difference matrix:

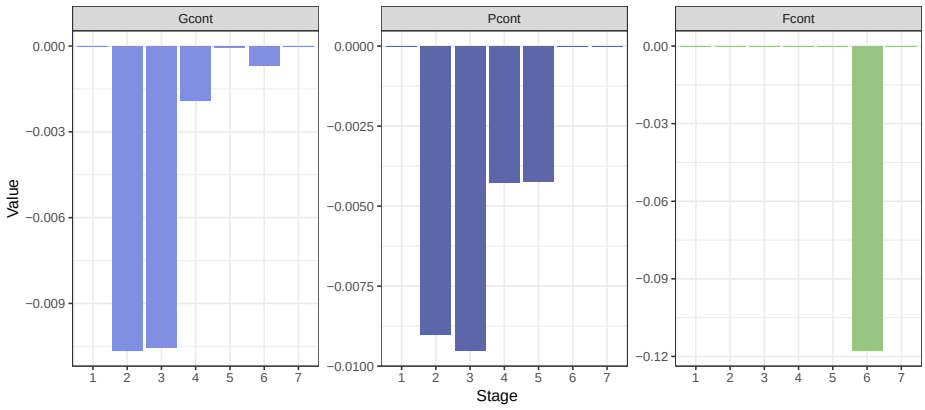
```
#Amat.isopod from previous exercise.
Halfway.matrix.isopod <- 0.5*(Amat.isopod.C+Amat.isopod)

Difference.matrix.isopod <- (Amat.isopod.C-Amat.isopod)
```

4.

Calculate sensitivity matrix and calculate / plot LTRE contributions:

```
#Modify element 6 of the G-vector, e.g. multiply it by 0.5
Sensitivity.Halfway.isopod <-
  sensitivity.matrix(Halfway.matrix.isopod)

LTRE.matrix.isopod <- sensitivity.matrix(Halfway.matrix.isopod)*
  Difference.matrix.isopod

#Extract components corresponding to G, P, and F across stages:
LTRE_F <- LTRE.matrix.isopod[1,]
LTRE_P <- c(diag(LTRE.matrix.isopod[2:7,1:6]),0)
LTRE_G <- diag(LTRE.matrix.isopod)


#Make a data frame for plotting:
```

```
LTREframe <- data.frame("Stage"=as.factor(1:7),
                        "Gcont"=LTRE_G,
                        "Pcont"=LTRE_P,
                        "Fcont"=LTRE_F)

#Long format:
LTREframe.long <- LTREframe %>%
  pivot_longer(c(Gcont, Pcont, Fcont),
               names_to = "Contribution",
               values_to = "Value")


#Plot with ggplot:
ggplot(LTREframe.long) +
  geom_col(aes(x=Stage, y=Value,
               fill=Contribution ))+
  scale_fill_manual(values=colors3)+
  facet_wrap(~factor(Contribution,
                     levels=c("Gcont", "Pcont", "Fcont")),
             scales="free")+
  theme_bw() +
  labs( x="Stage", y="Value")+
  theme(legend.position = "none" )
```



The contributions to $\lambda_C - \lambda_R \approx$ -0.1804 are highest from P and G of stage 2 and 3, and from fertility of stage 6. The sum of contributions is $\sum_{i,j} D_{ij} S^*_{ij} \approx$ -0.1687, a bit lower than the actual difference.

5.

LTRE is better if we want to understand the difference in λ in two different environments (or here, treatments), more specifically which parts of the projection matrix contribute to this difference. A sensitivity analysis applies to one

environment and tells us how λ would respond to small perturbations in each matrix element. The sensitivity analysis is prospective, while the LTRE analysis is retrospective [Caswell, 2001].

A.5 Chapter 5 exercises

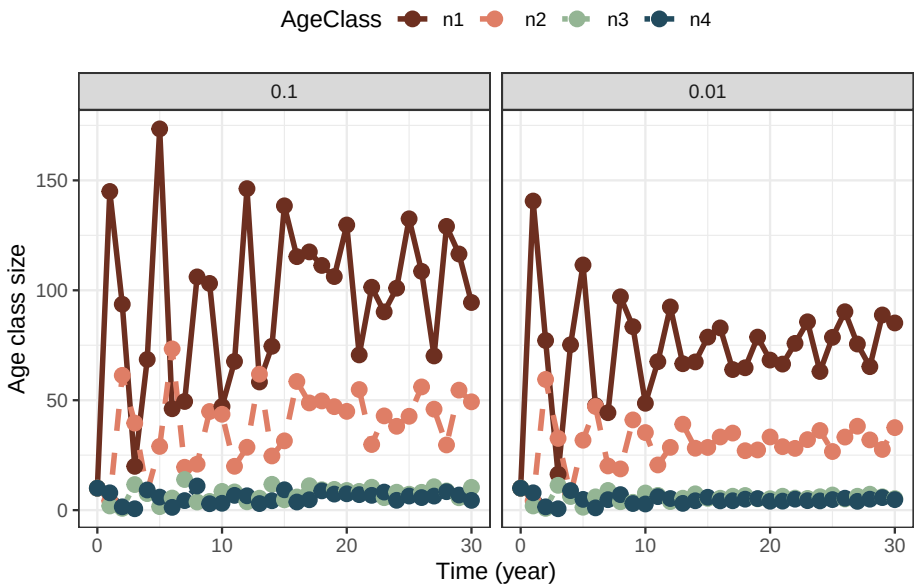
Exercise 5.1

1.

Add stochasticity to the fecundity coefficients, make plots:

```
#Project growth, variance 0.1
stochastic.plant <- projection.stochastic(
  Amats = Amat.array(MatA=APlant,
                     MatU=UPlant,
                     MatF=FPlant,
                     tmax=30,
                     VarF=rep(0.1, 4),
                     VarS=rep(0, 4)),
  n0=c(rep(10,4)))

stochastic.plant$varF <-  "0.1"

#Project growth, variance 0.01
stochastic.plant2 <- projection.stochastic(
  Amats = Amat.array(MatA=APlant,
                     MatU=UPlant,
                     MatF=FPlant,
                     tmax=30,
                     VarF=rep(0.01, 4),
                     VarS=rep(0, 4)),
  n0=c(rep(10,4)))

stochastic.plant2$varF <-  "0.01"


#Add the two:
stochastic.plant <- rbind(stochastic.plant, stochastic.plant2)


#Long format for plotting:
stochastic.plant.long <- stochastic.plant %>%
  pivot_longer(c(-time,-varF), names_to = "AgeClass",
               values_to = "Value")
```

```
#Plot:
ggplot(stochastic.plant.long) +
  geom_line(aes(x=time,  y=Value,
                col=AgeClass, linetype=AgeClass), lwd=1.2)+
  geom_point(aes(x=time, y=Value,
                 col=AgeClass), size=3)+
  theme_bw() +
   facet_wrap(~factor(varF,
                      levels=c("0.1", "0.01")))+
  scale_color_manual(values=colors4)+
  scale_linetype_manual(values=1:4)+
  labs( x="Time (year)", y="Age class size")  +
    theme(legend.position = "top" )
```



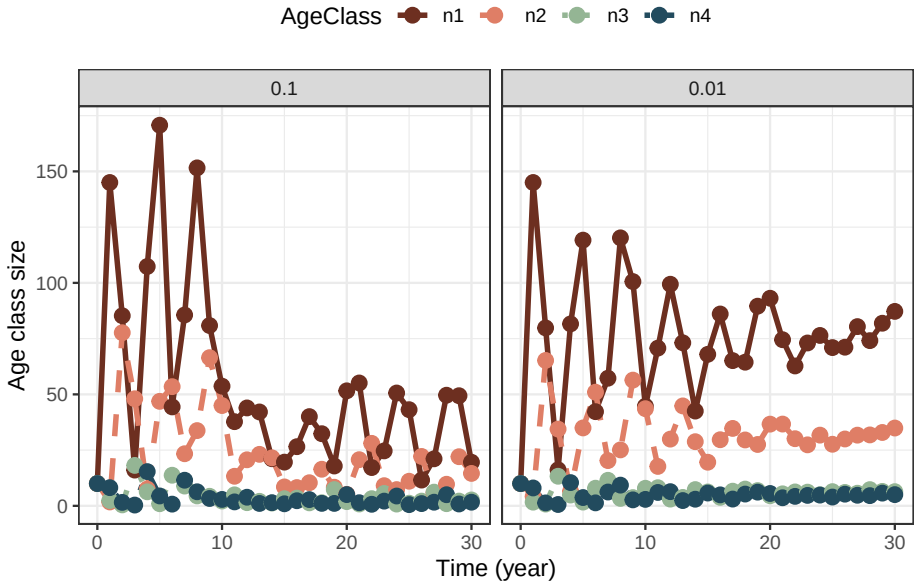
Variance in fertility increases the fluctuations in particular of the first age class. Note that we still have transient fluctuations in the beginning on top of the stochastic fluctuations. In the second case with lower variance in fertility, the variance in stage structure stabilizes at a lower value as well.

2.

Add stochasticity to the survival coefficients instead:

```
#Project growth, variance 0.1
stochastic.plant <- projection.stochastic(
  Amats = Amat.array(MatA=APlant,
                     MatU=UPlant,
                     MatF=FPlant,
```

```
                    tmax=30,
                    VarF=rep(0, 4),
                    VarS=rep(0.1, 4)),
  n0=c(rep(10,4)))

stochastic.plant$varS <-  "0.1"

#Project growth, variance 0.01
stochastic.plant2 <- projection.stochastic(
  Amats = Amat.array(MatA=APlant,
                MatU=UPlant,
                MatF=FPlant,
                tmax=30,
                VarF=rep(0, 4),
                VarS=rep(0.01, 4)),
  n0=c(rep(10,4)))

stochastic.plant2$varS <-  "0.01"


#Add the two:
stochastic.plant <- rbind(stochastic.plant,
                          stochastic.plant2)


#Long format for plotting:
stochastic.plant.long <- stochastic.plant %>%
  pivot_longer(c(-time,-varS),
                names_to = "AgeClass",
                values_to = "Value")

#Plot:
ggplot(stochastic.plant.long) +
  geom_line(aes(x=time, y=Value, col=AgeClass,
                linetype=AgeClass), lwd=1.2)+
  geom_point(aes(x=time, y=Value, col=AgeClass),
             size=3)+
  theme_bw() +
   facet_wrap(~factor(varS,
                      levels=c("0.1", "0.01")))+
  scale_color_manual(values=colors4)+
  scale_linetype_manual(values=1:4)+
  labs( x="Time (year)", y="Age class size")  +
   theme(legend.position = "top" )
```



We get quite similar patterns with variance on survival coefficients instead of fertility. This is because any fluctuations in one age class are carried on to the next, and to offspring. We always get more variation in the offspring class, because this is the largest class.

3. There are many potential ways a plant like this could show bet-hedging strategies, for instance:

- Conservative bet-hedging ('play it safe'): Reduce the mean fecundity to increase mean survival. This could mean that the plant might reduce annual variance in fitness, at the cost of not being able to maximize reproduction in good years.
- Diversifying bet-hedging ('don't put all eggs in one basket'): Spreading the risk among offspring (here seeds), either through different phenotypes, different spatial habitats, or different hatching times. The latter can be achieved through a seed bank, where seeds can lay dormant for many years before germinating.

A seed bank is a complicating factor to include in matrix models, but could be achieved by adding a new stage for the dormant seeds, with a certain probability of hatching each year. This would turn the model into a stage structured model.

One could argue that the plant life history already shows some bet-hedging since it reproduces at two ages, thus spreading the risk of reproducing at least among two years. Iteroparity can be interpreted as a bet-hedging strategy, but strictly speaking we would have to prove a cost to the arithmetic mean fitness to call it bet-hedging (e.g. lowering the maximum number of offspring that can be produced in good years).

Exercise 5.2

1.

Calculate λ for each habitat:

```
#Create the Leslie matrices
AmatPoor <- Create.Amat(Svec=c(.8,rep(.82,4)),
                        Fvec=c(0, 0.1, 0.16, 0.2, 0.2) )

AmatGood <- Create.Amat(Svec=c(.8,rep(.82,4)),
                        Fvec=c(0, 0.096, 0.16, 0.224, 0.32)  )


#Calculate lambda:

lambda.good <- uvlambda(AmatGood)$lam
lambda.poor <- uvlambda(AmatPoor)$lam

lambda.habitats <- data.frame("lambda.good"=lambda.good, "lambda.poor"=lambda.poor)

lambda.habitats
```

```
##   lambda.good lambda.poor
## 1     1.01092   0.9682986
```

We see that $\lambda > 1$ in the good habitat, and $\lambda < 1$ in the poor habitat. This calculation assumes no connection between them, treating each as a separate population (what would hypothetically happen to each over time with no migration between them).

2.

Generate sequence of matrices returned as array:

```
ArrayFunction <- function(Agood=AmatGood,
                          Apoor=AmatPoor,
                          time=30){
  #Generate sequence of good or bad habitat years:
  random.sequence <- sample(c("Good", "Poor"),
                            time,
                            replace=TRUE)
  #Generate empty array:
  k <- dim(Agood)[1]
  A.array <- array(NA, dim=c(k,k,time))
  #Fill in right matrix depending on the sequence
  for(i in 1:time){
    if(random.sequence[i]=="Good"){
      A.array[,,i] <- Agood
```

```
    }
    else {
       A.array[,,i] <- Apoor
    }
  }
  A.array
}

#Test:
ArrayFunction(time=3)
```

```
## , , 1
##
##      [,1]  [,2] [,3]  [,4] [,5]
## [1,]  0.0 0.096 0.16 0.224 0.32
## [2,]  0.8 0.000 0.00 0.000 0.00
## [3,]  0.0 0.820 0.00 0.000 0.00
## [4,]  0.0 0.000 0.82 0.000 0.00
## [5,]  0.0 0.000 0.00 0.820 0.82
##
## , , 2
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.0 0.10 0.16 0.20 0.20
## [2,]  0.8 0.00 0.00 0.00 0.00
## [3,]  0.0 0.82 0.00 0.00 0.00
## [4,]  0.0 0.00 0.82 0.00 0.00
## [5,]  0.0 0.00 0.00 0.82 0.82
##
## , , 3
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.0 0.10 0.16 0.20 0.20
## [2,]  0.8 0.00 0.00 0.00 0.00
## [3,]  0.0 0.82 0.00 0.00 0.00
## [4,]  0.0 0.00 0.82 0.00 0.00
## [5,]  0.0 0.00 0.00 0.82 0.82
```

3.

The mean growth rate corresponds (in this case) to the growth rate λ_m of the mean matrix:

```
MeanMatrix <- (AmatPoor+AmatGood)/2
r.mean <- log(uvlambda(MeanMatrix)$lambda)
```

Stochastic growth rate:

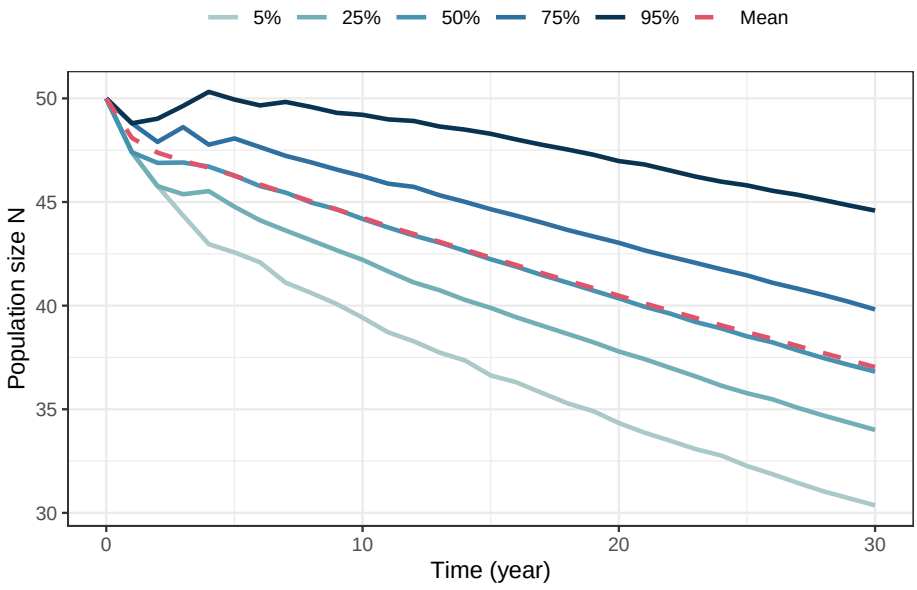
```
AMATS<-ArrayFunction(time=10000)
rs.gull <- EstimateSGR(Amats = AMATS, n0=rep(10,5))
```

For the gulls the mean growth rate is $r \approx$ -0.0089, while the stochastic growth rate is $r_s \approx$ -0.0092. Both are negative, the stochastic growth rate sligtly lower.

To make a quantile plot we perform a large number of stochastic simulations as described in the chapter and shown for the songbird example, after modifying the `nsim.stochastic` function to include the new random process of generating an array of matrices:

```
nsim.stochastic.gull <- function(Agood=AmatGood,
                                 Apoor=AmatPoor,
                                 tmax=30,
                                 nsim=1000,
                                 n0=rep(10,5)){
  frame <-  data.frame("Time"=0:tmax)
  for(m in 1:nsim){
    AMATS <- ArrayFunction(Agood=Agood,
                           Apoor=Apoor,
                           time=tmax)
    projectN <- apply(projection.stochastic(
      Amats=AMATS,  n0=n0)[,-1],1,sum)
    frame <- cbind(frame, projectN)
     names(frame)[m+1] <- paste("Rep",m,sep="")
  }
  frame
}

#Apply function to get 10 0000 realizations (can take some time)
simgull_Large <- nsim.stochastic.gull(nsim=10000)

#Get the 5, 25, 50, 75, and 95 percentiles:
PercentilesGull  <- t(apply(t(simgull_Large[,-1]),
                            2,
                            quantile,
                            probs=c(.05, .25, .5, .75,.95)))


#Store percentiles in data frame
GullPercentiles <-
  as.data.frame(cbind("Time"=0:(dim(PercentilesGull)[1]-1),
                                        PercentilesGull))

GullPercentiles$Mean <- apply(t(simgull_Large[,-1]),
                              2,
```

```
                                mean)

#Make long format for plotting
GullPercentiles.long <- GullPercentiles %>%
  pivot_longer(c(-Time),
               names_to = "Percentile", values_to = "N")

#Make sure the order of the percentiles is correct
GullPercentiles.long$Percentile <-
  factor(GullPercentiles.long$Percentile,
         levels=c("5%", "25%", "50%", "75%", "95%","Mean"))

ggplot(GullPercentiles.long) +
  geom_line(aes(x=Time, y=N, col=Percentile,
                linetype=Percentile), lwd=1)+
  scale_color_manual(values=c(colors5,2))+
  scale_linetype_manual(values=c(rep(1,5),2))+
  theme_bw() +
   theme(legend.position = "top",
         legend.title = element_blank())+
  labs( x="Time (year)", y="Population size N") +
   guides(color = guide_legend(nrow = 1, byrow = TRUE))
```



4.

First we add the probability argument to the function generating a random sequence of projection matrices:

```
ArrayFunction <- function(Agood=AmatGood,
                          Apoor=AmatPoor,
                          time=30,
                          prob=c(.5,.5)){
  #Generate sequence of good or bad habitat years:
  random.sequence <- sample(c("Good", "Poor"),
                            time,
                            replace=TRUE,
                            prob=prob)
  #Generate empty array:
  k <- dim(Agood)[1]
  A.array <- array(NA, dim=c(k,k,time))
  #Fill in right matrix depending on the sequence
  for(i in 1:time){
    if(random.sequence[i]=="Good"){
      A.array[,,i] <- Agood
    }
    else {
       A.array[,,i] <- Apoor
    }
  }
  A.array
}
```

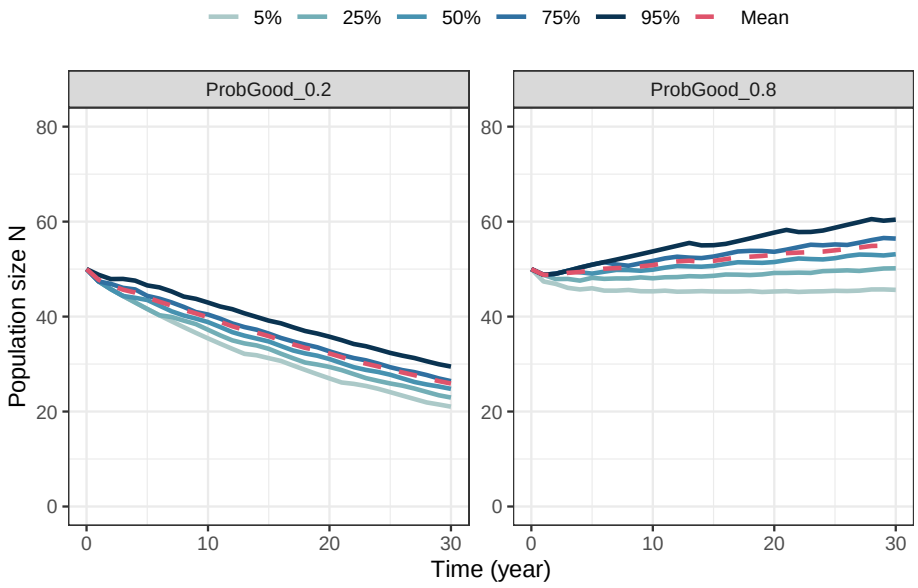
Then we modify the simulation function to include the argument

```
nsim.stochastic.gull <- function(Agood=AmatGood,
                                 Apoor=AmatPoor,
                                 tmax=30,
                                 nsim=1000,
                                 n0=rep(10,5),
                                 prob=c(.5,.5)){
  frame <-  data.frame("Time"=0:tmax)
  for(m in 1:nsim){
    AMATS <- ArrayFunction(Agood=Agood,
                           Apoor=Apoor,
                           time=tmax,
                           prob=prob)
    projectN <- apply(projection.stochastic(
      Amats=AMATS,  n0=n0)[,-1],1,sum)
    frame <- cbind(frame, projectN)
     names(frame)[m+1] <- paste("Rep",m,sep="")
  }
  frame
}
```

Now we can apply the function to compare situations where the probability of drawing a good environment is (e.g.) 0.8, versus 0.2:

```
#Most good environments
simgullG <- nsim.stochastic.gull(nsim=10000, prob=c(.8,.2))
#Most poor environments
simgullP <- nsim.stochastic.gull(nsim=10000, prob=c(.2,.8))

#Get the 5, 25, 50, 75, and 95 percentiles:
PercentilesG  <- t(apply(t(simgullG[,-1]),
                         2,
                         quantile,
                         probs=c(.05, .25, .5, .75,.95)))
PercentilesP  <- t(apply(t(simgullP[,-1]),
                         2,
                         quantile,
                         probs=c(.05, .25, .5, .75,.95)))

#Store percentiles in data frame
GPercentiles <-
  as.data.frame(cbind("Time"=0:(dim(PercentilesG)[1]-1),
                                    PercentilesG))
GPercentiles$Mean <- apply(t(PercentilesG[,-1]),
                           2,
                           mean)

PPercentiles <-
  as.data.frame(cbind("Time"=0:(dim(PercentilesP)[1]-1),
                                    PercentilesP))
PPercentiles$Mean <- apply(t(PercentilesP[,-1]),
                           2,
                           mean)

#Add probability value
GPercentiles$ProbGood <- "ProbGood_0.8"
PPercentiles$ProbGood <- "ProbGood_0.2"

#Join:

PercentilesBoth <- rbind(GPercentiles, PPercentiles)

#Make long format for plotting
PercentilesBoth.long <- PercentilesBoth %>%
  pivot_longer(c(-Time,-ProbGood),
               names_to = "Percentile", values_to = "N")
```

```
#Make sure the order of the percentiles is correct
PercentilesBoth.long$Percentile <-
  factor(PercentilesBoth.long$Percentile,
         levels=c("5%", "25%", "50%", "75%", "95%","Mean"))

ggplot(PercentilesBoth.long) +
  geom_line(aes(x=Time, y=N, col=Percentile,
                linetype=Percentile), lwd=1)+
  scale_color_manual(values=c(colors5,2))+
   scale_linetype_manual(values=c(rep(1,5),2))+
  theme_bw() +
  ylim(0,80)+
  facet_wrap(vars(ProbGood),ncol=2,
             scales="free")+
   theme(legend.position = "top",
         legend.title = element_blank() )+
  labs( x="Time (year)", y="Population size N") +
   guides(color = guide_legend(nrow = 1, byrow = TRUE))
```



In the case where the probability of using the good environment increases (to the right), population growth is higher (perhaps not surprising).

Appendix B

R code collection

This appendix includes a collection of the R functions used in the compendium.

An RData-file containing all the functions as well as relevant R objects for the songbird example can be downloaded here:

To import this file into R, use `load("CompendiumCode.RData")` (after saving the file in your R working directory folder). Then all functions and objects should be visible in the 'Environment' tab.

B.1 Life tables

B.1.1 Net reproductive rate R_0

```
R0.lifetable <- function(lx, mx){
  sum(lx*mx)
}
```

B.1.2 Euler-Lotka

Part of the method to solve the equation for λ using `uniroot()` (see main text).

```
EulerLotka <- function(lambda, x, lx, mx){
  sum(lambda^(-x)*lx*mx)-1
}
```

B.2 Matrix population models

B.2.1 Create Leslie matrix

Build Leslie matrix from vectors of survival and fertility coefficients (need to have same length). If the last survival coefficient is not zero, individuals can accumulate in the final class.

```
Create.Amat <- function(Svec, Fvec){
  k <- length(Svec)
  MatA<- matrix(0,k,k)
  MatA[1,] <- Fvec
  for(i in 1:(k-1)){
    MatA[i+1,i] <- Svec[i]
  }
  MatA[k,k] <- Svec[k]
  MatA
  }
```

B.2.2 Decompose U and F

Decompose the matrices **U** and **F** based on equation to obtain the transition matrix **G** vector of survival probability **s** (from **U**), and the transition matrix **Q** the fertility vector **f** (from **F**)

```
DecomposeU <- function(MatU){
  k<-dim(MatU)[1]
  if(is.na(MatU[k,k])){
    MatU[k,k]<-0
  }
  Svec <- apply(MatU,2,sum)
  MatG <- t(t(MatU)/Svec)
  for(i in 1:k){
    if(Svec[i]==0){
      MatG[,i] <- 0
      MatG[i,i] <- 1
    }
  }
 list("Gmat"=MatG, "Survival"=Svec)
}
```

```
DecomposeF <- function(MatF){
    k <- dim(MatF)[1]
    Fvec <- apply(MatF,2,sum)
    MatQ <- matrix(0,k,k)
```

```
    for(i in 1:k){
        if(Fvec[i]==0){
            MatQ[1,i]<-1

        }
        else{
            MatQ[,i]<-MatF[,i]/Fvec[i]
        }
    }
 list("Qmat"=MatQ, "Fertility"=Fvec)
}
```

B.2.3 Population projection

Project size of stages over time given a projection matrix and an initial population vector. Return as data frame.

```
projection <- function(MatA, Tmax = 50, n0){
  k <- dim(MatA)[1]
  if(length(n0)!=k){
      warning("n0 should have length
              k corresponding to
              number of stages in Amat")
      }
  Nmat <- matrix(NA, nrow= Tmax+1, ncol=k)
  cnames <- paste("n", 1:k, sep="")
  timesteps <- 0:Tmax
  Nmat[1,] = n0
  for(i in 2:(Tmax+1)){
    Nmat[i,] = MatA%*%Nmat[i-1,]
  }
  frame <- data.frame(timesteps, Nmat)
  colnames(frame) <- c("time", cnames)
  frame
}
```

B.2.4 Long-term growth rate, stable structure and reproductive value

For a projection matrix $\mathbf{A}$, the function returns the asymptotic growth rate λ, a vector describing the stable stage structure $\mathbf{u}$, and a vector describing the stage-specific reproductive values $\mathbf{v}'$. NB: We scale the reproductive values so that $\mathbf{v}'\mathbf{u} = 1$.

```
uvlambda <- function(MatA){
 ev <- eigen(MatA)
 tev <- eigen(t(MatA))
 lmax <- which.max(Re(ev$values))
 U <- ev$vectors
 V <- tev$vectors
 u <- as.matrix(abs(Re(U[, lmax]))/
                  sum(abs(Re(U[, lmax]))))
 u <- u/(sum(u)) #scale u
 v <- as.matrix(abs(Re(V[, lmax])))
 v <- v/sum(u*v ) #scale v
 v <- t(ifelse(u*v <= 0, 0, v))
 return(list("lambda"=max(Re(ev$values)),
             "u"=u,
             "v"=v))
}
```

B.2.5 Life expectancy

For a projection matrix MatU, the function returns the vector of life expectancies for given stages. Life expectancy at the first stage is the first element of this vector.

```
 LifeExpectancy <- function(MatU){
  k<-dim(MatU)[1]
  if(is.na(MatU[k,k])){
    MatU[k,k]<-0
  }
  uDim=dim(MatU)[1]
  N = solve(diag(uDim[1])-MatU)
  colSums(N)
}
```

B.2.6 Lifetime reproduction

The function returns the mean lifetime reproduction R_0, based on the matrices **A** and **U**. and the vector of expected remaining lifetime reproduction by stage. If all offspring are born into the first stage, the first element also corresponds to R_0.

```
R0function  <-  function(MatU, MatF){
  k <- dim(MatU)[[1]]
  Rmat <- MatF%*%solve(diag(1, k, k)-MatU)
  Rvec <- apply(Rmat,2,sum)
```

```
  R0 <- uvlambda(MatA=Rmat)$lam
  return(list("R0"= R0, "Rvec"= Rvec))
}
```

B.2.7 Generation time

For a projection matrix MatA and a reproduction matrix MatF, the function returns the generation time estimated as the mean age of mothers in a population with stable structure.

```
GenTime  <-  function(MatA, FMat){
  res <- uvlambda(MatA=MatA)
  lam <- res$lam
  u <- res$u
  v <- res$v
  lam/(v%*%FMat%*%u)
}
```

B.2.8 Sensitivities and elasticities

For a given projection matrix **A**, the function `sensitivity.matrix()` returns the matrix of sensitivities of λ to each corresponding element of MatA, and the function `elasticity.matrix()` returns the elasticities. The argument 'zeroes' can be set to FALSE to return also the sensitivities / elasticities corresponding to elements of MatA that are zero.

```
sensitivity.matrix <- function(MatA, zeroes=T){
  res <- uvlambda(MatA=MatA)
  sensmat <- t(res$u%*%res$v)
  if (zeroes==T){
    sensmat <- ifelse (MatA==0, 0, sensmat)
  }
 sensmat
}
```

```
elasticity.matrix <- function(MatA, zeroes=T){
  res <- uvlambda(MatA=MatA)
  sensmat <- t(res$u%*%res$v)
  if (zeroes==T){
    sensmat <- ifelse (MatA==0, 0, sensmat)
  }
 sensmat*MatA/res$lam
}
```

B.3 Stochastic models

B.3.1 Arithmetic and geometric mean

The function calculates the arithmetic and geometric mean of a sequence of annual population growth rates Λ_t with a given mean and standard deviation. It returns a data frame with time steps, the generated random sequence of annual growth rates Λ_t, and the two means.

```
AGMeans <- function(Tmax=100, mean.lambda=1.05, sd.lambda=.3){
  #Use lognormal distribution to avoid negative values
  Lambdavec <- rlnorm(n=Tmax,
                      log(mean.lambda)-
                        .5*log(1+ sd.lambda^2/
                                 (mean.lambda^2)),
                      sdlog=sqrt(log(1+sd.lambda^2/
                                       (mean.lambda^2))))
   geomLambda <- exp(mean(log(Lambdavec)))
   meanLambda <- mean(Lambdavec)
  data.frame("Time"=0:(length(Lambdavec)-1),
             "Lambda"=Lambdavec,
             "Geometric"=geomLambda,
             "Arithmetic"=meanLambda)
}
```

B.3.2 Simulations in Lewontin-Cohen model

Function to generate one realization of the model, given initial population size, mean and standard deviation of the annual growth rates Λ_t:

```
LC <- function(Tmax=30,
               n0=100,
               mean.lambda=1.05,
               sd.lambda=0.1){
  stochastic.lambdas <- rnorm(Tmax,
                              mean=mean.lambda,
                              sd=sd.lambda)
  Nvec <- rep(NA, Tmax+1)
  Nvec[1] <- n0
  for(i in 1:Tmax){
    Nvec[i+1] <- stochastic.lambdas[i]*Nvec[i]
  }
  data.frame("Time"=0:Tmax, "N"=Nvec)
}
```

Function to generate a chosen number (`nsim`) of realizations of the Lewontin-Cohen model, returned in a data frame.

```
nsimLC <- function(nsim=100,
                   Tmax=30,
                   n0=100,
                   mean.lambda=1.05,
                   sd.lambda=0.1){
  frame <-  data.frame("Time"=0:Tmax)
  for(j in 1:nsim){
  stochastic.lambdas <- rnorm(Tmax,
                              mean=mean.lambda,
                              sd=sd.lambda)
  Nvec <- rep(NA, Tmax+1)
  Nvec[1] <- n0
  for(i in 1:Tmax){
    Nvec[i+1] <- stochastic.lambdas[i]*Nvec[i]
  }
  frame <- cbind(frame, Nvec)
  names(frame)[j+1] <- paste("N",j,sep="")
  }
  frame
}
```

B.3.3 Simulations of stochastic structured model

B.3.3.1 Generate sequence of projection matrices

The function below returns a time series of stochastic projection matrices as an array, for a model assuming stochasticity in survival coefficients and fertility coefficients only, and no correlation between matrix elements (can be modified to include such correlation). Fertility is modeled with a log-link (normally distributed on log-scale), while survival is modeled with a log-log-link (normally distribued on log-log-scale). The mean (constant) environment is determined by the input matrices **A**, **U** and **F**. The input vectors `VarF` and `VarS` specify the variances of fertility and survival on their respective scales.

```
Amat.array <- function(MatA,
                       MatU,
                       MatF,
                       tmax=30,
                       VarF=rep(0.03, 3),
                       VarS=rep(0.03, 3)){
  #Split survival/transition matrix:
  SplitU <- DecomposeU(MatU)
```

```
  #Split fertility matrix:
  SplitF <- DecomposeF(MatF)
   #Transition matrix:
  Gmat <- SplitU$Gmat
   #Survival vector:
  Svec <- SplitU$Survival
  #Define beta (log-log link function):
  Beta.S <- -log(-log(Svec))
  #Offspring transition matrix:
  Qmat <- SplitF$Qmat
  #Fertility vector:
  Fvec <- SplitF$Fertility
  #Define beta (log link function):
  Beta.F <- log(Fvec)

  #variance covariance matrix
  SigMat <- diag(c(VarS, VarF))
  #Draw random (scaled) values for S and F:
  SFTime <- mvrnorm(tmax, mu=c(Beta.S, Beta.F), Sigma=SigMat)
  #Number of matrix classes
  k <- length(Svec)
  #Rescaled survival probabilities (loglog link):
  SvecTime<-   exp(-exp(-SFTime[,1:k]))
  #Rescaled fertilities  (log link):
  FvecTime<- exp(SFTime[,(k+1):(2*k)])
#Build  projection matrix for each time step:
  A.array <- array(NA,dim=c(k,k,tmax))
  for (i in 1:tmax){
    Smat <- diag(SvecTime[i,])
    Fmat <- diag(FvecTime[i,])
    A.array[,,i] <-   Gmat%*%Smat+Qmat%*%Fmat
  }
return("Amats" = A.array)
}
```

B.3.3.2 Stochastic population projection

The function below returns a data frame with the size of each stage over time, for a given array of projection matrices (third dimension should correspond to time) and initial population vector `n0`.

```
projection.stochastic <-function(Amats, n0){
    Tmax <- dim(Amats)[3]
    k <- dim(Amats)[1]
    if(length(n0)!=k){
```

```
      warning("n0 should have length k
              corresponding to number of
              stages in the Amats object")
      }
    Nmat <- matrix(NA,nrow= Tmax+1, ncol=k)
    Nmat[1,] <- n0
  cnames <- paste("n", 1:k, sep="")
  timesteps <- 0:Tmax
    for(i in 2:(Tmax+1)){
        Nmat[i,] <- Amats[,,i-1] %*% Nmat[i-1,]
        }
  frame <- data.frame(timesteps,Nmat)
  colnames(frame) <- c("time", cnames)
  frame
}
```

The following function generates a series of stochastic matrices `nsim` times, each series of length `Tmax`, and returns the projected total population size over time for each series, in a data frame. Other input variables are the constant matrices and variance vectors, and initial population vector.

```
nsim.stochastic <- function(nsim=100,
                            tmax=30,
                            n0=rep(10,3),
                            MatA,
                            MatU,
                            MatF,
                            VarF,
                            VarS){
  frame <-  data.frame("Time"=0:tmax)
  for(m in 1:nsim){
  AMATS <- Amat.array(MatA=MatA,
                      MatU=MatU,
                      MatF=MatF,
                      VarF=VarF,
                      VarS=VarS,
                      tmax=tmax)
   projectN <- apply(projection.stochastic(Amats=AMATS,
                                           n0=n0)[,-1],1,sum)
  frame <- cbind(frame, projectN)
  names(frame)[m+1] <- paste("Rep",m,sep="")
  }
  frame
}
```

B.3.4 Estimate stochastic growth rate

The following code can be used to calculate the stochastic growth rate following the definition of equation (5.10), given one sequence of projection matrices. The cutoff argument removes the first time steps (corresponding to cutoff number).

```
EstimateSGR <- function(Amats, n0, cutoff=20){
  Tmax <- dim(Amats)[3]
    if(cutoff>=Tmax){
      warning("cutoff must be lower
              than the number of
              matrices in Amats")
      }
  project <- projection.stochastic(Amats=Amats, n0=n0)
  Ntot <- apply(project[,-1],1,sum)
  logN <- log(Ntot[cutoff:length(Ntot)])
  TM <- length(logN)
  (logN[TM]-logN[1])/TM
}
```

B.4 Parameters and matrices for the songbird example

Copy paste if you want to look at example code without running all the previous code.

```
#------------------------------
#Life table parameters
#------------------------------
x <- 0:3
mx <- c(0, 0, 3, 6)
lx <- c(1, 0.2, 0.18, 0.108)
px <- c(.2, .9, .6, 0)

#------------------------------
#Life table calculations
#------------------------------
TC_Bird <- TC.lifetable(x,  lx, mx)
R0_Bird <- R0.lifetable(lx, mx)

#------------------------------
#Apply uniroot to find lambda
#(EulerLotka defined among funcions)
#------------------------------
```

```
lambdabird <- uniroot(EulerLotka,
                      x=x,
                      lx=lx,
                      mx=mx,
                      interval=c(.1,2))$root

#------------------------------
#Projection matrices
#------------------------------
Abird.pre <- Create.Amat(
  Svec <- c(.9, .6, 0),
  Fvec= c(0, .6, 1.2))
Abird.post <- Create.Amat(
  Svec <- c(.2, .9, 0),
  Fvec= c(0, 2.7, 3.6))

#------------------------------
#Projection
#------------------------------
Pop.bird.pre <- projection(
  MatA=Abird.pre,
  n0=rep(10,3),
  Tmax=30)

Pop.bird.post <- projection(
  MatA=Abird.post,
  n0=rep(10,3),
  Tmax=30)
```

Bibliography

C. Bieber and T. Ruf. Population dynamics in wild boar *Sus scrofa*: ecology, elasticity of growth rate and implications for the management of pulsed resource consumers. *J. Appl. Ecol.*, 42:1203–1213, 2005.

H. Caswell, C. de Vries, N. Hartemink, G. Roth, and S. F. van Daalen. Age x stage-classified demographic analysis: a comprehensive approach. *Ecol. Monogr.*, 88:560–584, 2018.

Hal Caswell. *Matrix population models*. Sinauer Associates, Sunderland, Massachusetts, 2nd edition, 2001.

Hal Caswell. Extrinsic mortality and the evolution of senescence. *Trends Ecol. Evol.*, 22:173–174, 2007.

D. Cohen. Optimizing reproduction in a randomly varying environment. *Journal of Theoretical Biology*, 12:119–129, 1966.

Lamont C. Cole. The population consequences of life history phenomena. *The quarterly review of biology*, 29:103–137, 1954.

M. R. Easterling, Stephen P. Ellner, and P. M. Dixon. Size-specific sensitivity: applying a new structured population model. *Ecology*, 81:694–708, 2000.

Stephen P. Ellner, Dylan Z. Childs, and Mark Rees. *Data-driven Modelling of Structured Populations*. Lecture Notes on Mathematical Modelling in the Life Sciences. Springer International Publishing Switzerland, 2016.

Ronald Aylmer Fisher. *The genetical theory of natural selection*. Clarendon Press, Oxford, 1930.

W. D. Hamilton. The moulding of senescence by natural selection. *J. Theor. Biol.*, 12:12–45, 1966.

Christine M. Hunter and Hal Caswell. The use of the vec-permutation matrix in spatial matrix population models. *Ecol. Model.*, 188:15–21, 2005.

J. E. Kammenga, C. A. M. van Gestel, and E. Hornung. Switching life-history sensitivities to stress in soil invertebrates. *Ecol. Appl.*, 11:226–238, 2001.

Bruce E. Kendall, M. Fujiwara, J. Diaz-Lopez, S. Schneider, J. Voigt, and S. Wiesner. Persistent problems in the construction of matrix population models. *Ecol. Model.*, 406:33–43, 2019.

W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics. *Proc. R. Soc. Lond. A*, 115:700–721, 1927.

David Lack. The significance of clutch size. *Ibis*, 89:302–352, 1947.

J. D. Lebreton. Demographic models for subdivided populations: The renewal equation approach. *Theor. Popul. Biol.*, 49:291–313, 1996.

L. P. Lefkovitch. The study of population growth in organisms grouped by stages. *Biometrics*, 21:1–18, 1965.

P. H. Leslie. On the use of matrices in certain population mathematics. *Biometrika*, 33:183–212, 1945.

P. H. Leslie. Some further notes on the use of matrices in population mathematics. *Biometrika*, 35:213–245, 1948.

R. C. Lewontin and D. Cohen. On population growth in a randomly varying environment. *Proc. Natl. Acad. Sci. U.S.A.*, 62:1056–1060, 1969.

Thomas Robert Malthus. *An essay on the principle of population.* J. Johnson, London, 1798.

Peter Brian Medawar. *An Unsolved Problem of Biology.* M. K. Lewis, London, 1952.

R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/.

J. Starrfelt and Hanna Kokko. Bet-hedging—a triple trade-off between means, variances and correlations. *Biol. Rev.*, 87:742–755, 2012.

Shripad Tuljapurkar. Population dynamics in variable environments. III. Evolutionary dynamics of r-selection. *Theor. Popul. Biol.*, 21:114–140, 1982.

Y. Vindenes, C. le Coeur, and H. Caswell. Introduction to matrix population models. In R. Salguero-Gomez and M. Gamelon, editors, *Demographic methods across the tree of life*, pages 163–179. Oxford University Press, Oxford, UK., 2020.

Maarten J. Wensink, H. Caswell, and Annette Baudisch. The rarity of survival to old age does not drive the evolution of senescence. *J. Evol. Biol.*, 44:5–10, 2017.

Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson,

Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686.

George C. Williams. Pleiotropy, natural selection, and the evolution of senescence. *Evolution*, 11:398–411, 1957.