Invasibility Analysis

3.1 Introduction

An alternative approach to that used in the last chapter is invasibility analysis, which consists of asking if a clone displaying an alternate life history can invade a resident population. While one could compare results for markedly different life histories, in general, invasibility analysis has been used to locate the optimal combinations of parameter values rather than qualitatively different life histories. As with the "Fisherian" optimality approach, sexual reproduction is ignored. Invasibility analysis is used extensively, and is most useful, when fitness is density-dependent and there is age- or stage-structuring in the model. The method can handle stable, cyclical, and chaotic population dynamics. In this section I first consider the general structure of age- and stage-structured models and then describe the two general approaches of invasibility analysis, namely pairwise-invasibility and multiple-invasibility analysis. For all that you ever wanted to know about matrix population models see Caswell (2002).

3.1.1 Age- or stage-structured models

Consider the life table shown in Table 3.1.

Table 3.1 Calculation of age-specific survival probabilities and fertilities for the Leslie matrix.

Age *x*	l_x	m_x	**Post-breeding census**		**Pre-breeding census**	
			$S_x = \frac{l_x}{l_{x-1}}$	$F_x = S_x m_x$	$S_x = \frac{l_{x+1}}{l_x}$	$F_x = S_x m_x$
0	1	0	–	–	0.8	0
1	0.80	1	0.80	0.8	0.4	0.4
2	0.20	3	0.40	1.2	0.25	0.75
3	0.05	4	0.25	1.0	0	0
4	0.00	1	0.00	0.00	–	–

Using the post-breeding census, which is an assumption for the models discussed in Chapter 2, the number of individuals entering age 1 at time $t + 1$ is given by

$$\begin{aligned} n_{1,t+1} &= S_1 m_1 n_{1,t} + S_2 m_2 n_{2,t} + S_3 m_3 n_{3,t} + S_4 m_4 n_{4,t} \\ &= (0.8)(1) n_{1,t} + (0.4)(3) n_{2,t} + (0.25)(4) n_{3,t} + (0) n_{4,t} \\ &= 0.8 n_{1,t} + 1.2 n_{2,t} + 1 n_{3,t} + 0 n_{4,t} \end{aligned} \tag{3.1}$$

where $n_{i,t}$ is the number in age class i at time t. The number surviving from t to $t + 1$ is given by

$$\begin{aligned} n_{2,t+1} &= S_1 n_{1,t} &= 0.8 n_{1,t} \\ n_{3,t+1} &= S_2 n_{2,t} &= 0.4 n_{2,t} \\ n_{4,t+1} &= S_3 n_{3,t} &= 0.25 n_{3,t} \end{aligned} \tag{3.2}$$

This set of equations can be represented in matrix format as

$$\begin{pmatrix} n_{1,t+1} \\ n_{2,t+1} \\ n_{3,t+1} \\ n_{4,t+1} \end{pmatrix} = \begin{pmatrix} F_1 & F_2 & F_3 & F_4 \\ S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \end{pmatrix} \begin{pmatrix} n_{1,t} \\ n_{2,t} \\ n_{3,t} \\ n_{4,t} \end{pmatrix} \tag{3.3}$$

If a pre-breeding census is assumed, then $S_x = l_{x+1}/l_x$ (Caswell 1989, p. 12). The matrix can be written in shorthand as

$$\mathbf{n}_{t+1} = \mathbf{A}\mathbf{n}_t \tag{3.4}$$

The matrix **A** is known as the **Leslie matrix** after the ecologist who first introduced it into the population biology literature (Leslie 1945). The advantage of using the matrix notation is that there are well-defined rules for manipulating matrices, particularly for matrix multiplication. From an evolutionary biologist's point of view the important feature of this matrix is that the population rate of increase at a stable age distribution, λ, is given by the first **eigenvalue** of the matrix. This value is readily obtained in R or MATLAB. For example, in the life history specified by equations (3.1) and (3.2) the Leslie matrix is

$$\mathbf{A} = \begin{pmatrix} 0.8 & 1.2 & 1.0 & 0 \\ 0.8 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{pmatrix} \tag{3.5}$$

which can be entered using R as

```
Leslie.matrix <- matrix(c(0.8, 1.2, 1.0,  0,
                          0.8, 0.0, 0.0,  0,
                          0.0, 0.4, 0.0,  0,
                          0.0, 0.0, 0.25, 0 ),4,4, byrow=TRUE)
```

where, for ease of writing, I have aligned the columns. The eigenvalues and eigenvectors can be obtained with the command `eigen` (R) or `eig` (MATLAB).

In R the appropriate eigenvalue can be drawn from the list with the suffix `$values[1]`. Thus the following commands in R,

```
Eigen.data <- eigen(Leslie.matrix)
Eigen.data$values[1]   # Get first eigenvalue
```

gives 1.5516.

Equation (3.3) defines the change in population size after one generation. If the initial population consists of a single gravid female the population size in the next generation is given by

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix} = \begin{pmatrix} 0.8 & 1.2 & 1.0 & 0 \\ 0.8 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.8 \\ 0 \\ 0 \end{pmatrix} \tag{3.6}$$

Matrix multiplication in R is coded by %*%, thus

```
n <- Leslie.matrix%*%n
```

gives the multiplication shown in equation (3.6). Progressive application of matrix multiplication produces the population projection. The following coding calculates and plots the trajectories of the individual cohorts (ages 1–4) and the total population size. Additionally, the observed rate of increase, given by N_{t+1}/N_t, and the instantaneous rate of increase, r, given by $\log_e(N_{t+1}/N_t)$ are plotted.

```
  rm(list=ls()) # Clear workspace
  Leslie.matrix <- matrix(c(0.8, 1.2, 1.0, 0,
                            0.8, 0.0, 0.0, 0,
                            0.0, 0.4, 0.0, 0,
                            0.0, 0.0, 0.25,0),4,4, byrow=TRUE)
  Eigen.data <- eigen( Leslie.matrix)
  Lambda     <- Eigen.data$values[1]   # Get first eigenvalue
  Maxgen     <- 12                     # Number of generations
                                         simulation runs
  n          <- c(1,0,0,0)             # Initial population
# Pre-assign matrix to hold cohort number and total population size
  Pop        <- matrix(0,Maxgen,5)
  Pop[1,]    <- c(n[1:4], sum(n))      # Store initial population
# Pre-assign storage for observed lambda
  Obs.lambda   <- matrix(0,Maxgen,5)
  for ( Igen in 2:Maxgen)            # Iterate over generations
  {
  n <- Leslie.matrix%*%n             # Apply matrix multiplication
  Pop[Igen,1:4]<- n[1:4]          # Store cohorts
  Pop[Igen,5] <- sum(n)           # Store total population size

  Obs.lambda[Igen,] <- Pop[Igen,]/Pop[Igen-1,] # Store observed
                                                 lambda
  }                               # End of Igen loop
# Print out observed lambda in last generation and ratio
  print(c(Obs.lambda[Maxgen], Obs.lambda[Maxgen]/Lambda))
  par(mfrow=c(2,2))               # Make 2x2 layout of plots
  Generation <- seq(from=1, to=Maxgen)   # Vector of generation
                                           number
# Plot population and cohort trajectories
  ymin <- min(Pop); ymax <- max(Pop) # get minimum and maximum pop
                                       sizes
  plot(   Generation,    Pop[,1],    type='l',ylim=c(ymin,ymax),
ylab='Population and cohort sizes') # Cohort 1
  for( i in 2:4) {lines(Generation, Pop[,i]) } # Cohorts 2-4
  lines(Generation, Pop[,5], lty=2) # Total population
# Plot log of population and cohort trajectories
# Log zero is undefined so remove these
  x <- matrix(Pop,length(Pop),1)   # Convert to one dimensional
                                     matrix
  ymin <- min(log(x[x!=0]))   # minimum log value
  ymax <- max(log(Pop))        # get minimum and maximum pop sizes
  plot( Generation, log(Pop[,1]), type='l', ylim=c(ymin,ymax),
ylab='log Sizes')
  for(i in 2:4) {lines(Generation, log(Pop[,i]))}
  lines(Generation, log(Pop[,5]), lty=2)     # Total population
# Plot Observed lambdas
  plot(Generation, Obs.lambda[,1], type='l', ylab='Lambda')
  for( i in 2:4) {lines(Generation, Obs.lambda[,i])}
  lines(Generation, Obs.lambda[,5], lty=2)  # Total population
# Plot observed r
  plot(Generation, log(Obs.lambda[,1]), type='l', ylab='r')
  for( i in 2:4) {lines(Generation, log(Obs.lambda[,i]))}
  lines(Generation, log(Obs.lambda[,5]), lty=2)
                                               # Total population
```

OUTPUT: (Figure 3.1)

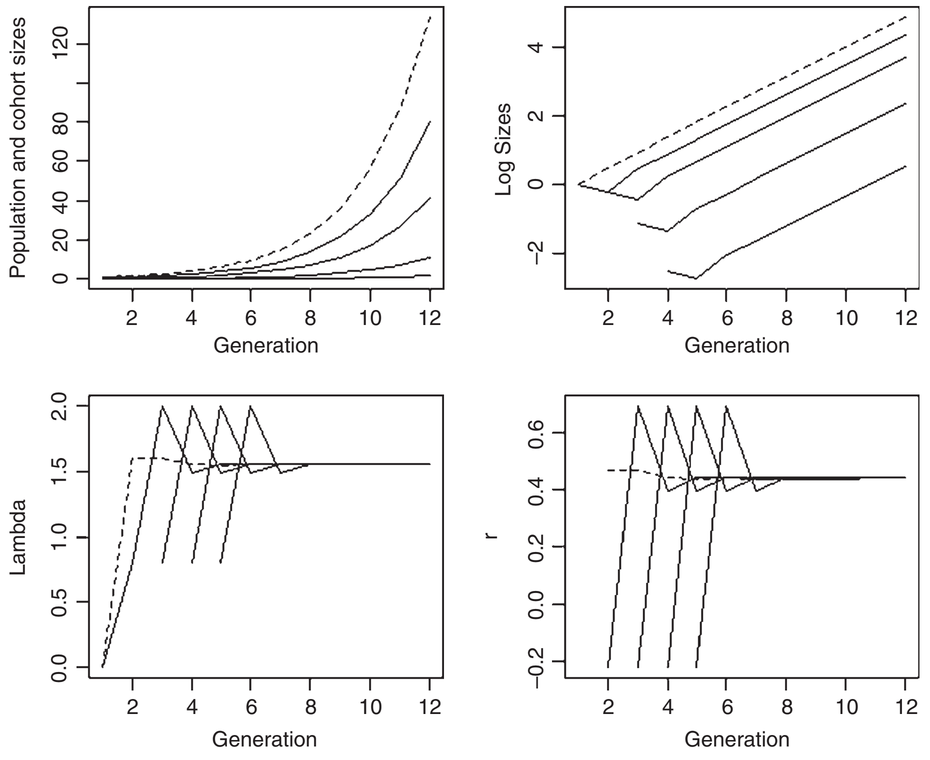

Figure 3.1 Trajectories of cohort (solid lines) and population sizes (dotted line) and the observed values of λ and r.

```
> print(c(Obs.lambda[Maxgen], Obs.lambda[Maxgen]/Lambda))
[1] 1.5516186+0i 0.9999971+0i
```

The simulation shows that the population quickly reaches a stable age distribution, shown by the linearity of the plot of log(population or cohort size) on time and the constancy of the observed λ (Figure 3.1).

3.1.2 Modeling evolution using the Leslie matrix

Because the population quickly reaches a stable age distribution and there is no density-dependence the methods presented in Chapter 2 can be used to analyse models defined by a Leslie matrix. However, because of the ease with which λ or r ($= \log_e\lambda$) is calculated from a Leslie matrix, a matrix approach can sometimes be a more easily programmed method than those used in Chapter 2. Scenario 1 gives an example of finding the optimal life history using the Leslie matrix compared to the approach used in Chapter 2.

3.1.3 Stage-structured models

In many cases a life cycle is better classified according to stages rather than ages: for example, the transition from juvenile to adult is probably more frequently dependent on passing some size-threshold than a particular age. Suppose we have a population in which maturity depends upon reaching a minimum size, after which there are two adult stages. The two adult stages differ and passage from one to another is also size dependent (e.g., in the first adult stage males might be too small to compete for territories and adopt a satellite strategy. Note that in this case the symbol F refers to reproductive success). The three transition equations are

$$\begin{aligned} n_{1,t+1} &= P_1 n_{1,t} + F_2 n_{2,t} + F_3 n_{3,t} \\ n_{2,t+1} &= S_1 n_{1,t} + P_2 n_{2,t} \\ n_{3,t+1} &= S_2 n_{2,t} \end{aligned} \tag{3.7}$$

where P_{i} is the surviving proportion that remain in the ith stage and S_{i} is the proportion that pass from stage i and survive to the next stage. These equations can be converted into the matrix

$$\begin{pmatrix} n_{1,t+1} \\ n_{2,t+1} \\ n_{3,t+1} \end{pmatrix} = \begin{pmatrix} P_1 & F_2 & F_3 \\ S_1 & P_2 & 0 \\ 0 & S_2 & 0 \end{pmatrix} \begin{pmatrix} n_{1,t} \\ n_{2,t} \\ n_{3,t} \end{pmatrix} \tag{3.8}$$

There is no fundamental mathematical difference between age and stage-structured models and the latter can be analyzed using the "Fisherian" optimality approach. Difficulties arise when fitness is density-dependent, a topic to which we now turn.

3.1.4 Adding density-dependence

The Leslie matrix or its stage-based analogue can be readily modified to accommodate density-dependent effects. There are many ways that a density-dependent effect can be entered, for example, fertility might only be affected or survival or both. Only one age class might be affected or the effect spread over several or all age classes. Two common functions are the Beverton–Holt function and the Ricker function (both named after the fisheries biologists who suggested it). The Beverton–Holt function is compensatory in that it progresses smoothly to an asymptotic value, whereas the Ricker function is overcompensatory in that for some portion of the curve N_{t+1} is less than N_t. The standard forms of these two models for an unstructured population are

$$\begin{aligned} N_{t+1} &= N_t \frac{c_1}{1 + c_2 N_t} \quad \text{Beverton} - \text{Holt} \\ N_{t+1} &= N_t \alpha e^{-\beta N_t} \qquad \text{Ricker} \end{aligned} \tag{3.9}$$

The Beverton–Holt model asymptotes at an equilibrium population, whereas the Ricker model can equilibrate, cycle, or show chaotic behavior (Figure 3.2). In applying these functions the population size terms immediately adjacent to the

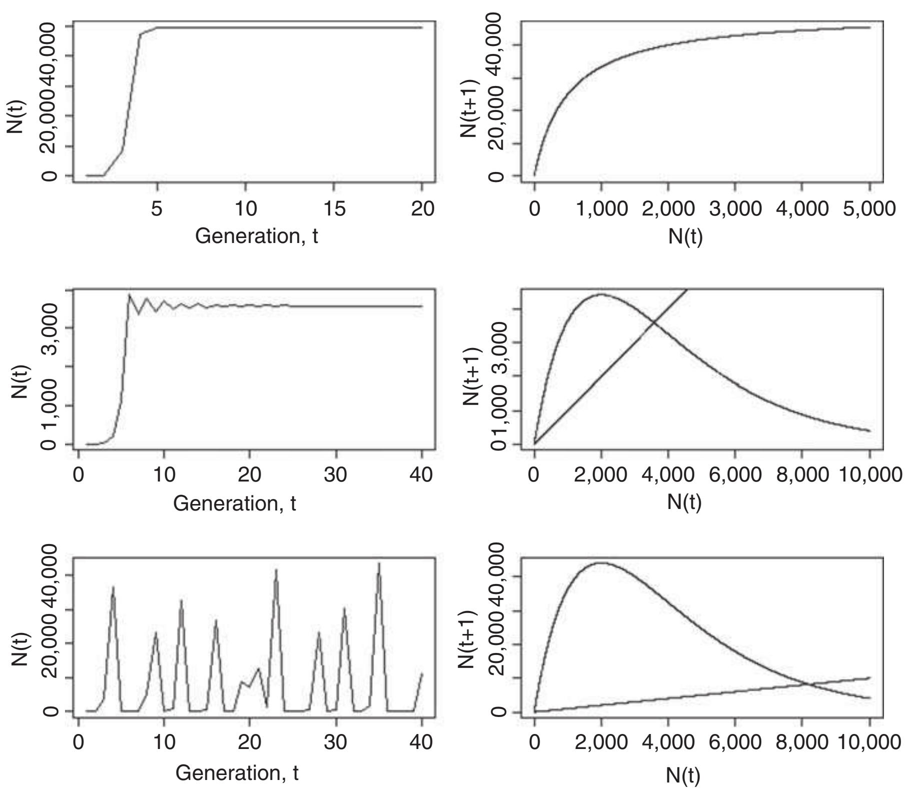

Figure 3.2 Examples of population trajectories for the Beverton–Holt (first row) and Ricker models. Depending on parameter values, the Ricker model may reach a stable equilibrium (second row), or show cyclical behavior (not shown) or chaotic behavior (third row). Plots on the right show the change in population size as a function of the previous population. The R coding to produce these plots is as follows:

```
  rm(list=ls())              #  Clear workspace
  par(mfrow=c(3,2))          #  Divide page into 6 panels
  BH.FUNCTION       <-   function(n,c1,c2)  {c1/(1+c2*n)}
  RICKER.FUNCTION   <-  function(n, ALPHA, BETA)  {ALPHA*exp
(-BETA*n)}
###################  MAIN PROGRAM  ###################
##########  Beverton Holt function  ##########
  c1  <-  100;   c2 <- 2*10^-3  # B-H parameters
# Plot N(t) on t
  Maxgen  <- 20; N.t  <- matrix(0,Maxgen); N.t[1]   <-  1
  for ( i in 2:Maxgen)
  {N.t[i] <- N.t[i-1]*BH.FUNCTION(N.t[i-1], c1,c2)}
  plot(seq(from=1, to=Maxgen), N.t, xlab = 'Generation, t', ylab
='N(t)',type='l')
# Plot N(t+1) on N(t)
  MaxN       <- 5000; N.t   <- matrix(seq(from=1, to=MaxN))
  N.tplus1   <- N.t*apply(N.t,1,BH.FUNCTION, c1,c2)
```

(cont'd)

Fig 3.2 (cont'd)

```
  plot(N.t, N.tplus1, xlab = 'N(t)', ylab='N(t+1)', type='l')
##########  Ricker function  ##########
    ALPHA   <-c(6, 60);   BETA   <- .0005    # Parameter values
# Plot N(t) on t for 2 values of ALPHA
    Maxgen  <- 40
    for (j in 1:2)
{
  N.t       <- matrix(0,Maxgen,1);    N.t[1] <- 1
  for   ( i in 2:Maxgen)
  {N.t[i]<- N.t[i-1]*RICKER.FUNCTION(N.t[i-1], ALPHA[j], BETA)}
  plot(seq(from=1, to=Maxgen), N.t, xlab = 'Generation, t', ylab
='N(t)', type='l')
# Plot N(t+1) on N(t)
  MaxN   <- 10000;     N.t   <- matrix(seq(from=1, to=MaxN))
  N.tplus1   <- N.t*apply(N.t, 1, RICKER.FUNCTION, ALPHA[j],BETA)
  plot(N.t, N.tplus1, xlab='N(t)', ylab='N(t+1)', type='l')
  lines(N.t, N.t)
  } # End of j loop
```

equality sign are replaced by fertility and/or survival terms. Thus if fertility in the previously described Leslie matrix is modified by a Ricker density dependent function that affects all ages we have

$$A_t = \begin{pmatrix} 0.8\alpha e^{-\beta N_t} & 1.2\alpha e^{-\beta N_t} & 1.0\alpha e^{-\beta N_t} & 0\alpha e^{-\beta N_t} \\ 0.8 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{pmatrix} \tag{3.10}$$

where N_t may be the total population size or some particular set of ages (see example below). How one introduces the density-dependent function is determined by the biological assumptions. Similarly, the particular density-dependent function is a function of the particular biological scenario envisaged. If one wishes to do a general analysis, both functions, with a range of parameter values, should be tried. Another suggested density-dependent function is the Usher function:

$$\frac{1}{1+e^{aN+b}} \tag{3.11}$$

which produces a sigmoidal growth curve. Benton and Grant (1999) modified this function to produce a gradual or sudden onset of density-dependence:

$$\begin{aligned} &\frac{1}{1+e^{1.25bN}} - 50,000b \quad \text{gradual onset} \\ &\frac{1}{1+e^{12.5bN}} - 500,000b \quad \text{sudden onset} \end{aligned} \tag{3.12}$$

where $b = 2 \times 10^{-5}$. None of the above equations are sacrosanct and in the absence of detailed information any function that produces a density-dependent effect might be tried. In general, the Beverton–Holt and Ricker functions do cover a wide range of behaviors and are reasonable functions to use.

A simple example of a stage structured model that includes density-dependence is that for *Tribolium spp.* proposed by Dennis et al. (1995) and further analyzed by Grant and Benton (2003). The life cycle of the beetle is divided into three stages, larval, pupal, and adult with transitions between stages governed by the following assumptions:

1. The number of larvae at time $t + 1$, L_{t+1} is determined by the number of adults at time t, A_t, the rate at which eggs are cannibalized by adults, $c_{A.\text{eggs}}$, and the rate of cannibalization by the larvae, $c_{L.\text{eggs}}$. These effects can be modeled by a Ricker function.

$$L_{t+1} = bA_t e^{-(c_{A.\text{eggs}}A_t + c_{L.\text{eggs}}L_t)} \tag{3.13}$$

where b is a constant.

2. The number of pupae that survive to time $t + 1$ is

$$P_{t+1} = L_t S_L \tag{3.14}$$

where S_L is the survival probability of non-cannibalized larvae.

3. The number of adults is a function of the number of pupae that are cannibalized by the adults (a Ricker function) and the survival of adults (S_A):

$$A_{t+1} = P_t e^{-c_{A.\text{pupae}}A_t} + A_t S_A \tag{3.15}$$

These three equations can be written in matrix form as

$$\begin{pmatrix} L_{t+1} \\ P_{t+1} \\ A_{t+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & be^{-(c_{A.\text{eggs}}A_t + c_{L.\text{eggs}}L_t)} \\ S_L & 0 & 0 \\ 0 & e^{-c_{A.\text{pupae}}A_t} & S_A \end{pmatrix} \begin{pmatrix} L_t \\ P_t \\ A_t \end{pmatrix} \tag{3.16}$$

3.1.5 Estimating fitness

If density-dependence is not a function of the trait of interest and the population is stable then an appropriate measure of fitness is R_0, which will generally be much easier to evaluate than using an invasibility approach (see Scenario 2). The operational definition of fitness for invasibility analysis is the ability of a novel clone (the invader) to invade a resident population. However, this does mean that the invader will replace the resident population as it could coexist with the resident. The fitness of the invader is the long-term growth rate of the invader population, which can be equated to the dominant Lyapunov exponent of the matrix. In most cases relevant to this book this exponent, also called the invasion exponent, has to be estimated by simulation. Two approaches for determining the equilibrium set of trait variables are pairwise invasibility analysis and multiple invasibility analysis.

3.1.6 Pairwise invasibility analysis

This is a graphical method that identifies putative Evolutionarily Stable Strategies (ESS) on a surface comprising the set of combinations of resident and invader trait values. There are four possible outcomes, diagrammed in Figures 3.3 and 3.4. The x-axis is the set of trait values for the resident and the y-axis is the same set of trait values representing the trait values of the invader. For each combination we estimate the long-term growth rate of the invader. The hypothetical long-term growth rate of the invader in the stationary resident population is given by the dominant Lyapunov exponent, called by Rand et al. (1994) the invasion exponent, ϑ:

$$\vartheta = \lim_{t \to \infty} \frac{1}{t} \ln \frac{N_t}{N_0} \tag{3.17}$$

Because of the small population size of the invader population, the invasion exponent can be estimated by assuming that the invader population will either increase or decrease exponentially (at least measured over a sufficient time period):

$$\begin{aligned} N_{t+1} &= N_0 e^{r_{\text{invader}} t} = N_0 \lambda^t_{\text{invader}} \\ \ln N_{t+1} &= \ln N_0 + t \ \ln \lambda_{\text{invader}} \end{aligned} \tag{3.18}$$

Thus after some specified number of iterations the growth rate of the invader population, ϑ, can be estimated from a linear regression of log(invader population size) on generation.

Two contour lines are shown on the invasibility plots of Figure 3.3. Both lines denote the set of combinations at which the growth rate of the invader is zero. Obviously when the parameter value of the invader is the same as that of the resident then the invader will neither increase nor decrease: this is the $x = y$ line shown in the plots. Now consider a trait combination that lies very close to the origin but above the line of equality: at this point the growth rate of the invader is positive and it increases in frequency and eventually becomes the resident population. For a combination that lies in the upper right of the plots the growth rate of the invader is negative and it cannot penetrate the resident population. Thus at some combinations other than $x = y$ the growth rate of the invader must equal that of the resident population. The point at which this second zero isocline crosses the line of equality is the putative ESS. Several such points could exist or there could be zero isoclines that do not intersect the line of equality (e.g., see White et al. [2006]). Whether the putative ESS is a stable ESS (termed a **convergence stable ESS**) or an unstable equilibrium depends on the shape of the second zero isocline: if the slope of the second isocline is greater than 90° as measured in relation to the x and y-axes (see top plots in Figure 3.3) the intersection is an ESS, otherwise the equilibrium is unstable and subject to invasion (bottom panels of Figure 3.3). The plot on the left of Figure 3.3 shows a case in which the putative ESS is a convergence stable ESS, while that on the right shows a case in which the intersection defines an unstable equilibrium termed an **evolutionary branching**

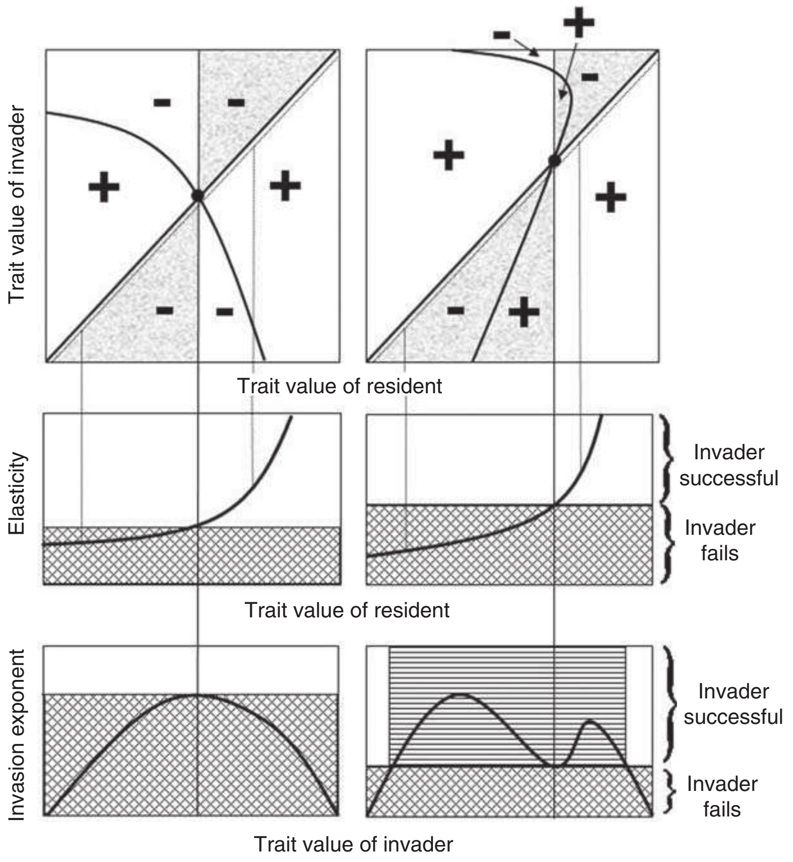

Figure 3.3 Hypothetical examples of pairwise invasibility plots (top panels) in which there is convergence but not necessarily an ESS. The panels on the left show a convergence stable ESS and those on the right show an evolutionary branching point. A "+" denotes a positive long-term growth rate of the invader population (i.e., invasion successful) and a "–" indicates a negative long-term growth rate (i.e., invasion unsuccessful). The dotted lines paralleling the x = y line indicate the values used in the elasticity analysis and the vertical dotted lines show examples of the elasticity values obtained at those points. The shaded areas indicate the zones that are relevant for plotting the invasion exponent of the invader against the putative ESS value of the resident as shown in the bottom panels. Panels below the first row show the elasticity analyses. In the middle panels the trait value of the invader is set at some fraction slightly smaller than 1 (e.g., 0.995) of the trait value of the resident. This analysis is used to determine the putative ESS value. In the bottom panels the trait value of the resident is set at the putative ESS. This analysis determines if the putative ESS value is resistant to invasion. The cross-hatched areas indicate those resident–invader combinations which lead to extinction of the invader. The horizontal hatched areas indicate trait values for which invasion occurs when the resident population is at its putative ESS. In the left-hand column there are no values for which invasion is successful when the resident population is at the putative ESS, whereas in the right-hand plot there are values for which invasion is successful.

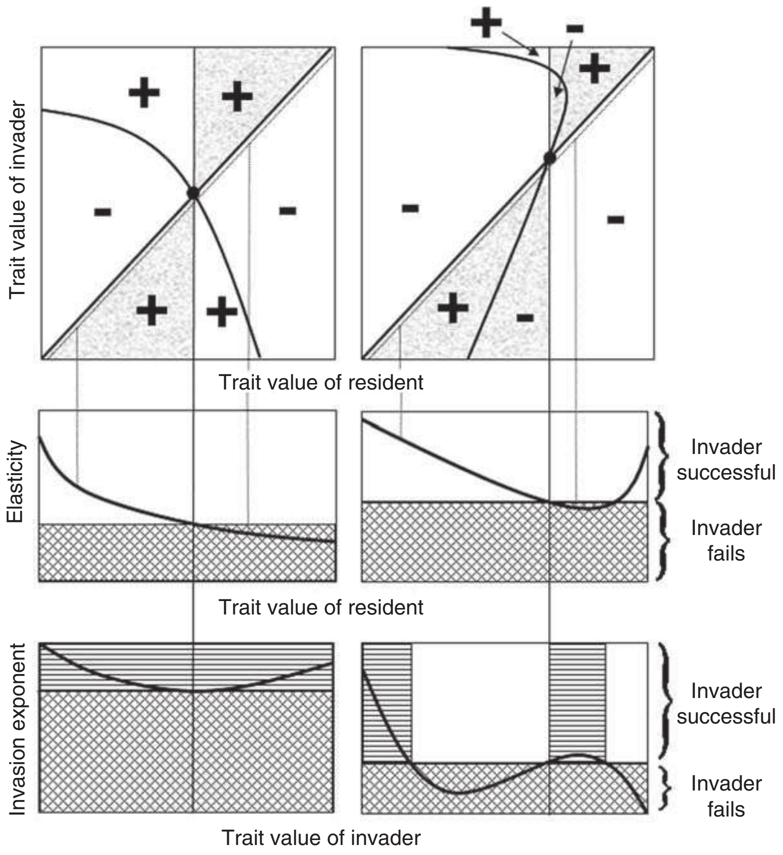

Figure 3.4 Hypothetical examples of pairwise invasibility plots (top panels) in which there is neither convergence nor a stable ESS. The panels on the right show an invasible repellor and those on the left show a Garden-of-Eden ESS. A "+" denotes a positive long-term growth rate of the invader population (i.e., invasion successful) and a "−" indicates a negative long-term growth rate (i.e., invasion unsuccessful). The dotted lines paralleling the x = y line indicate the values used in the elasticity analysis and the vertical dotted lines show examples of the elasticity values obtained at those points. The shaded areas indicate the zones that are relevant for plotting the invasion exponent of the invader against the putative ESS value of the resident as shown in the bottom panels. Panels below the first row show the elasticity analyses. In the middle panels the trait value of the invader is set at some fraction slightly smaller than 1 (e.g., 0.995) of the trait value of the resident. This analysis is used to determine the putative ESS value. In the bottom panels the trait value of the resident is set at the putative ESS. This analysis determines if the putative ESS value is resistant to invasion. The cross-hatched areas indicate those resident-invader combinations which lead to extinction of the invader. The horizontal hatched areas indicate trait values for which invasion is indicated by both the analysis of elasticity with respect to the trait value of the resident (middle panels) and with respect to the trait value of the invader (bottom panels). In both cases there are combinations from both the elasticity and invasion exponent plots for which invasion is successful.

point. In theory the ESS is not resistant to mutants and polymorphisms will occur (however, see Scenario 5 of Chapter 4, in which the "unstable" ESS of Scenario 3 of this chapter is stable when parameters are inherited according to a quantitative genetic model). The plot of elasticity versus the trait value of the resident shows that there is convergence but the invasion exponent plotted against the trait value of the invader shows that invasion is possible in the rightmost scenario. There are two other possible pairwise invasibility plots, obtained if the areas defining the positive and negative growth of the invader are reversed (Figure 3.4). In both cases the elasticity plotted against the trait value of the resident shows that there is no convergence and the invasion exponent versus the trait value of the invader shows that invasion is possible in both scenarios. The scenario on the left is termed an **invasibility repellor** and that on the right a **Garden-of-Eden ESS**.

Suppose the trait under study, say X, can reasonably range from X_{min} to X_{max}. To produce a pairwise invasibility plot we proceed as follows:

Step 1: Divide X_{min} to X_{max}. into N_{inc} increments. This set of values will be applied to residents and invaders: for example, in R

```
X.Resident <- seq(from=X.min, to=X.max, length=N.inc)
X.Invader <- X.residents
```

Step 2: Create the set of all combinations for resident and invader types. This can be done using the R function `expand.grid`

```
Combinations <- expand.grid(X.Resident, X.Invader)
```

Step 3: For each combination calculate the population growth rate of the invader entering a resident population. If this growth rate is positive then the invader trait value has a higher fitness than the resident trait value. The calculation of the invader growth rate will typically be estimated by calling some function, say `POP.DYNAMICS` that has the following elements in sequence:

a. The call to function `POP.DYNAMICS` passes the parameter value, in this case `ALPHA`, and the multiplier for the invader parameter value, in this case called `Coeff`. These two parameters could be passed as a vector of length 2 or, as done below, as separate elements.

```
POP.DYNAMICS   <- function(ALPHA, Coeff)
ALPHA.resident    <- ALPHA           # Alpha for resident
ALPHA.invader    <- ALPHA*Coeff      # Alpha for invader
```

b. Iteration of population growth of the resident population alone until it has passed any effects due to initial starting conditions (this does not necessarily mean that the population will be at equilibrium as it might exhibit cyclical or chaotic behavior or subject to environmental fluctuations). For example, suppose we run the resident-only time trace for 50 generations and the time trace after the invader is introduced for 300 generations. To hold the entire trace, which we might wish to do for later plotting, we need a matrix of 350 rows.

```
  Maxgen1      <- 50   # Generations when only resident present
  Maxgen2      <- 300  # Generations after invader introduced
  Tot.Gen <- Maxgen1+Maxgen2 # Total number of generations
  N.resident <- N.invader <- matrix(0,Tot.Gen)  # Allocate space
  N.resident[1]  <- 1             # Initial number of resident
  N.invader[Maxgen1] <- 1         # Initial number of invader
  for (Igen in 2:Maxgen1)         # Iterate over only resident
{
  N <- N.resident[Igen-1]         # For typing convenience
  N.resident[Igen] <- DD.FUNCTION (ALPHA.resident, N, N)
} # End of resident only period
```

The density-dependent function, `DD.FUNCTION`, takes the parameter value, the population size of the focal type (resident or invader), and the total population size. In the present case these are the same. An example of the density dependent function, which is that used in Scenario 3, is

```
  DD.FUNCTION  <-  function(ALPHA,N1,N2)   #  Density-dependence
                                              function
{
  BETA         <-  ALPHA*0.001             # Set value of beta
  N  <-  N1*ALPHA*exp(-BETA*N2)      # New population size
  return(N)
}  # End  of  DD.FUNCTION
```

c. Introduction of a single invader into the resident population, which should be large so that the initial population size of the invader has no significant effect on the density-dependent effect. Note that now the call to the density-dependent function passes the total population size as the third element. Although the number of invaders should be sufficiently small that they contribute insignificantly to the density-dependence I prefer to include their number in the total population size as it seems more biologically realistic.

```
# Now  add  invader
  J  <-  Maxgen1+1          # Starting generation of this period
  for (Igen  in  J:Tot.Gen) # Iterate after introduction of invader
{
  N.tot <-  N.resident[Igen-1]+ N.invader[Igen-1] # Total
                                                    popn size
  N.resident[Igen] <- DD.FUNCTION(ALPHA.resident, N.resident
[Igen-1], N.tot)
 N.invader[Igen] <- DD.FUNCTION(ALPHA.invader, N.invader
[Igen-1], N.tot)
} # End of invasion period
```

d. As noted above the hypothetical long-term growth rate of the invader in the stationary resident population is estimated by the slope of a linear regression of log (invader population size) on generation. Because of initial fluctuations in the invader population due to initial population composition, it may be necessary to ignore the first few generations (in the example coding below I ignore the first 10 generations). The number of generations that the model must be run to get an accurate estimate of the growth of the invader population will depend on the dynamics of the population – if there are fluctuations due to intrinsic population properties (e.g., a Ricker function) or environmental factors the number of generations may be large (e.g., 500) whereas if the model shows little fluctuation only 50 generations might be required. The appropriate number can be assessed by trial and error: lack of smoothness in the curves constructed from the simulated data will generally indicate an insufficient number of generations.

```
  Generation   <- seq(from=1, to=Tot.Gen) # Generation sequence
  N0   <- 10 + Maxgen1         # Starting point for regression
# Regression model
  Invasion.model <- lm(log(N.invader[N0:Tot.Gen])~ Generation
  [N0:Tot.Gen])
  Elasticity     <- Invasion.model$coeff[2] # Elasticity
  return(Elasticity)
} # End of POP.DYNAMICS function
```

e. The function passes the estimated growth rate of the invader to the main program. The growth rate of the invader is estimated for all combinations and the result converted into matrix form from which a contour plot can be constructed

```
  z          <- apply(Combinations,1,POP.DYNAMICS)
  z.matrix   <- matrix(z, N.inc, N.inc)    # Convert to a matrix
# Plot contours
  contour(X.Resident, X.Invader, z.matrix, xlab="Resident",
ylab="Invader")
```

Step 4: If there is an ESS there will be at least two relevant zero isoclines. The first is the line described by the equation `X.Resident = X.Invader` (obviously if the invader trait equals the resident trait it has the same fitness as the resident). Suppose there is a single ESS, this implies that there must be a second zero isocline that intersects the first (Figure 3.3). The two zero isoclines divide the plane into four quadrats as shown in Figure 3.3, where the shape of the second zero isocline will depend upon the details of the model.

Step 5: The putative ESS value can be read off the graph and its stability gauged from the isocline shapes.

While the above approach may demonstrate the existence of an equilibrium point it does not provide a ready means of determining the trait value. One way to examine the stability of the point and to numerically obtain its value is the elasticity approach of Grant (1997).

3.1.7 Elasticity analysis

To understand the mechanics of this method we need only consider how to estimate the putative ESS from the pairwise invasibility plot. Suppose we start at $X_{\min}$ for the resident population and set the trait value of the invader at some value slightly below that of the resident population (dotted lines in Figures 3.3 and 3.4) say 0.995 $X_{\min}$, which is the value suggested by Benton and Grant (1999). At this point the growth rate of the invader population is negative and the invader cannot invade. We now sequentially advance the value of the resident trait value, increasing that of the invader by the same proportion of the resident value as before (the dotted lines shown in Figures 3.3 and 3.4). When the resident trait value exceeds the putative ESS value the sign of the invader growth rate changes we have passed the putative ESS point and we have fixed the ESS value within the limits set by the increments by which we increased the invader trait value (middle panels in Figures 3.3 and 3.4)

The growth rate of the invader population in the above situation is called the elasticity (see Chapter 1) of the invasion exponent to a change in the resident trait value. Provided that the elasticity is a monotonic function of the trait value, as shown in Figure 3.3, the point at which the elasticity is zero, which is the ESS value, can be found using a numerical search routine such as `uniroot`.

```
  Optimum <- uniroot(POP.DYNAMICS, interval=c(X.min,X.max),0.995)
  Best.E  <- Optimum$root   # Store the optimum reproductive effort
  print(Best.E)             # Print optimum E
```

As always it is good practice to use a graphical analysis to confirm the above answer:

```
  # Create plot of elasticity versus E
    N.int     <- 30         # Nos of increments
  # Create sequence of X from X.min to X.max in N.int increments
        X     <- matrix(seq(from=X.min, to=X.max, length=N.int),
N.int,1)
  # Create vector of elasticities using apply function
      Elasticity   <- apply(X, 1, POP.DYNAMICS, 0.995)
      plot(E, Elasticity, type='l') # Plot elasticity vs E
      lines(c(X.min,X.max), c(0,0)) # Add horizontal line at zero
```

As a final check we plot the invasion exponent of the invader (ϑ = long-term growth rate) relative to a resident population with the predicted optimum trait value. The sign of this is indicated by the shaded areas in Figures 3.3 and 3.4. If, as in the left-hand example shown in Figure 3.3, the predicted value is the ESS then all ϑ not equal to the ESS value should be negative and $\vartheta = 0$ at the ESS value. If the putative ESS is not resistant to invasion, as in the right-hand example of Figure 3.3

and the plots in Figure 3.4, the invasion exponent will not be negative for all values other than the putative ESS value (Figures 3.3, 3.4 and Scenario 6).

R CODE:

```
# Now plot Invasion exponent when resident is optimal
  Coeff             <- E/Best.E      # Coeff of invader DD function
  Invasion.exponent <- matrix(0,N.int,1) # pre-allocate space
# Iterate and calculate invasion exponent
# Note that a loop is used rather than apply because it is coefficient
  that is changing
  for (i in 1:N.int){ Invasion.exponent[i] <- POP.DYNAMICS
(Best.E, Coeff[i])
}
  plot(E, Invasion.exponent, ylab ='Invasion exponent', type='l')
  points(Best.E,0, cex=2) # Plot point at previously estimated
optimum E
```

In the scenarios that follow I have commenced the analysis by producing graphical output using pairwise invasibility analysis, but have placed on the graph the combination subsequently found with elasticity analysis.

3.1.8 Multiple invasibility analysis

An alternative approach that has been adopted is to introduce mutant clones at each generation into the population. This approach potentially permits the accumulation of multiple types in a population and thus demonstrates the existence of polymorphic populations but has the disadvantage that it is extremely computer intensive. The general approach is as follows:

1. We need to follow the sizes of cohorts with particular parameter values. There are two ways in which this can be accomplished. The first and simplest way is to turn the range of the parameter value into discrete units, for example, suppose the parameter, `X`, can vary from 2 to 15. This range can be divided into some specified number of intervals, say 50:

```
            X <- sequence(from=2, to=15, length=50)
```

The number of individuals in each class can be placed in a separate vector, or the two can be combined into a single matrix with the class values in the first column and the numbers in the second. Suppose we commence with a single individual in the middle of the range (more or less)

```
  Data     <- matrix(0, 50, 2)                      # Allocate apace
  Data[,1] <- sequence(from=2, to=15, length=50) # Set X values
  Data[25,2]  <- 1                               # Initial population
```

An alternate method is to generate types and follow them through time. The advantage of this alternate approach is that it permits the population to move to its ESS exactly. The disadvantage of this method is that it complicates the bookkeeping and it may be necessary at specified intervals to purge types that are in low numbers or the number of types to be kept track of will become exorbitant. This problem can be resolved in the former method by increasing the number of divisions though this will, of course, increase computational time. Because the only difference is one of bookkeeping I shall use only the former approach.

2. A density-dependent function must be specified. As an example, suppose that population size is determined by a Ricker function,

$$N_{t+1} = \alpha N_t e^{-\beta N_{\text{total}}} \tag{3.19}$$

in which there is a trade-off between the density-independent component α and the density dependent component β. This trade-off is actually specified by a positive relationship such as $\beta = 0.001\alpha$, which is used in Scenario 3. Coding for this function is as follows:

```
 DD.FUNCTION<- function(X, N.total) # Density-dependence function
{
# Set parameter values
   ALPHA <- X[1]                      # Set alpha
   N     <- X[2]                      # Population size for this
                                        alpha
   BETA <- ALPHA*0.001                # Set value of beta
   N <- N*ALPHA*exp(-BETA*N.total)    # New cohort size
   return(N)
} # End of function
```

3. New cohort sizes are generated by using the R function `apply`, providing it with the density-dependent function, `DD.FUNCTION`:

```
 N.total  <- sum(Data[,2])                     # Total population
                                                 size
 Data[,2] <- apply(Data,1,DD.FUNCTION, N.total) # New population
                                                   sizes
```

4. The above is enclosed within a loop that iterates over generations. After each generation new types are introduced at a low frequency. These are generally referred to as “mutations” but this assumes a biological scenario that can be misleading. The object of the analysis is to examine the placement of the optimal trait value, should it exist, and its stability. As written, the model assumes a clonal structure, which may apply to some organisms but generally not to the ones for which the analysis is supposed to apply. The assumption is that the results will apply in general to both clonal and sexual organisms. A comparison of Scenario 6 in this chapter with Scenario 5 of the next chapter, in which the same scenario is examined using a quantitative genetic perspective scenario, suggests that this assumption may, in some instances, be erroneous. Given this, I believe that it is better to regard the analysis not as a biological scenario but simply as a mathematical means of judging potential evolutionary history in the sense of movement to a single ESS, maintenance of polymorphisms or the existence of multiple equilibria. In the two examples presented in the subsequent scenarios I assume that a new type is introduced into the population at each generation: should this be judged too liberal, it is easy to alter the coding to make the introduction of a new type a probabilistic event (e.g., the type of "mutation" could be depend on the frequency of types already present in the population). Based on the assumption that a new type appears at each generation and is a random draw from all the possible types (this could also be changed such that the frequency distribution is, say, normal rather than uniform), coding is

```
  for (Igen in 1:Maxgen)
{
  N.total   <- sum(Data[,2])              # Total population size
  Data[,2]  <- apply(Data,1,DD.FUNCTION, N.total)    # New cohort
# Keep track of population size, mean trait value and SD of trait value
  Stats[Igen,2] <- sum(Data[,1]*Data[,2])/sum(Data[,2]) # Mean
  S          <- sum(Data[,2])                      # Popn size
  Stats[Igen,1]  <- S                               # Popn size
  SX1            <- sum(Data[,1]^2*Data[,2])
  SX2            <- (sum(Data[,1]*Data[,2]))^2/S
  Stats[Igen,3]  <- sqrt((SX1-SX2)/(S-1))       # SD of trait
# Introduce a mutant by picking a random integer between 1 and 50
  Mutant         <- ceiling(runif(1, min=0, max=50))
  Data[Mutant,2] <- Data[Mutant,2]+1        # Add mutant to class
} # End of Igen loop
```

In the above coding the program keeps track of the total population size, `Stats[Igen,1]`, the mean trait value, `Stats[Igen,2]`, and its standard deviation, `Stats[Igen,3]`. If there is a unique equilibrium the mean value should asymptote to this value and the standard deviation should equilibrate at a value determined by the difference between adjacent bins of the trait value (i.e., `Data[,1]`). If there are multiple equilibria the mean should fluctuate and the standard deviation should not reach a small limiting value. A plot of trait value class on population size (called a frequency polygon) is useful to provide a visual indication of the spread of the trait values.

Multiple invasibility analyses are given in Scenario 3, where there is a unique equilibrium and in Scenario 6, where the trait value fluctuates wildly.

3.2 Summary of scenarios

Scenario 1: Illustrates the use of the Leslie matrix to solve Scenario 5 of Chapter 2, in which there is no density-dependence and the population achieves a stable age distribution.

Scenario 2: Takes Scenario 1 and adds density-dependence that is independent of body size, which changes the fitness measure and thus the optimum body size.

Scenario 3: Considers a model in which population dynamics is governed by the Ricker function with a dependency between the components of the Ricker function.

Scenario 4: Gives another example of an age-structured model with density-dependence affecting the trait of interest. In this case the trait under study is the optimal reproductive effort.

Scenario 5: A stage-structured model in which the immature stage may delay moving into the adult stage. Depending on the proportion delaying maturity, the density dependent function can induce cyclical population dynamics which greatly affects the required number of generations that must be followed in the elasticity analysis.

Scenario 6: A model demonstrating the coexistence of multiple types in a population.

3.3 Scenario 1: Comparing approaches

To Illustrate and compare the approach used in Chapter 2 with that using a matrix modeling approach I shall use the example given in Scenario 5 of Chapter 2, with the change that a discrete time model rather than an integral model is used.

3.3.1 General assumptions

1. The organism is iteroparous.
2. Fecundity, F, increases with body size, x, which does not change after maturity (e.g., as in insect).
3. Survival, S, decreases with body size, x.
4. Fitness, W, is a function of fecundity and survival.

3.3.2 Mathematical assumptions

1. Maturity occurs at age 1 after which no further growth occurs.

2. Fecundity increases linearly with size at maturity, resulting in fecundity being a uniform function of age:

$$F_t = a_F + b_F x \tag{3.20}$$

3. The instantaneous rate of mortality increases linearly with the body size attained at age 1 and is constant per time unit. Under this assumption, survival to age t is given by

$$S_t = e^{-(a_s+b_s x)t} \tag{3.21}$$

4. Taking r to be the measure of fitness, the fitness function is given by the solution of the characteristic equation

$$\sum_{t=1}^{\infty} e^{-rt}(a_F + b_F x)e^{-(a_s+b_s x)t} = 1 \tag{3.22}$$

where the initial value of the summation is set at 1, as this is the age of first reproduction.

3.3.3 Solving using the methods of Chapter 2

The two exponents can be absorbed into a single term, giving

$$\sum_{t=1}^{\infty}(a_F + b_F x)e^{-(a_s+b_s x+r)t} = 1 \tag{3.23}$$

The above equation is a geometric series (see Section 2.5.2 for a discussion) and can be reduced to

$$\frac{(a_F + b_F x)e^{-(a_s+b_s x+r)}}{1 - e^{-(a_s+b_s x+r)}} = 1 \tag{3.24}$$

For convenience in the following derivation let $A = a_F + b_F x$ and $B = a_S + b_S x$ giving

$$\begin{aligned}
\frac{Ae^{-(B+r)}}{1 - e^{-(B+r)}} &= 1 \\
Ae^{-(B+r)} &= 1 - e^{-(B+r)} \\
e^{-(B+r)}(A+1) &= 1 \\
-(B+r) + \log_e(A+1) &= 0 \\
r &= \log_e(A+1) - B \\
&= \log_e(a_F + b_F x + 1) - (a_s + b_s x)
\end{aligned} \tag{3.25}$$

Thus we have an explicit expression for r as a function x (body size). Following the recipes given in Chapter 2 (e.g., Scenario 3, Section 2.5.2) the optimal body size is readily found. We define a function `RCALC` to calculate r using equation (3.25) and the call the R function `optimize` to locate the value of x at which fitness (r) is maximized:

```
  rm(list=ls())                            # Clear workspace
  RCALC <- function(x)                     # Function to calculate r
  Af <-0; Bf<-16; As<-1; Bs<-0.5           # parameter values
  r <- log(Af+Bf*x+1)-(As+Bs*x)           # r
  return(r)                                # return value
}
# Call optimize to find best x
  optimize(f=RCALC,interval= c(0.1,3), maximum=TRUE)$maximum
```

OUTPUT:

```
[1] 1.937494
```

If the fitness equation (3.22) cannot be resolved into a simple function of *r* it may be necessary to locate *r* by numerical means as done in Section 2.3.5. Note that this requires that we use a finite sum. Because of the rapid decline in survival with age only about 10 age classes are necessary: however, it is advisable to try several values to ensure that the result is not altered. Here I use 50, which gives essentially the same as answer as 10. Plotting *r* versus *x* (not shown) indicates that the optimum *x* lies between 1 and 3. This interval is passed to `optimize`.

```
  rm(list=ls())                            # Clear workspace
# Define function to sum characteristic eqn given r and x
  SUMMATION <- function(r,x)
{
  Maxage <- 50                          # Maximum age
  age    <- seq( from=1, to=Maxage)        # Vector of ages
  Af     <- 0; Bf <- 16; As <- 1 ; Bs <- 0.5 # Parameter values
  m <- rep(Af+Bf*x, times= Maxage)      # number of female offspring
  l <- exp(-(As+Bs*x)*age)              # Survival to age
  Sum    <- sum(exp(-r*age)*l*m)           # Characteristic eqn sum
  return(1-Sum)                         # Subtract 1 and return
}
# Define function to find r given x
  RCALC <- function(x){uniroot(SUMMATION, interval=c(-1,2),x)
$root}
# Calculate the best x by calling optimize, which calls RCALC
  optimize(f=RCALC,interval= c(0.1,3), maximum=TRUE)
```

OUTPUT:

```
[1] 1.937458
```

The result matches to three decimal places that obtained using the exact equation.

3.3.4 Solving using the eigenvalue of the Leslie matrix

The first task is to convert the life table specified by the model into a Leslie matrix. The coding is contained within the function `RCALC` which passes back the estimates *r*, calculated as the log of the first eigenvalue. This function is called by the R function `optimize`. Important points to note are

1. The age-specific survival (i.e., survival from age *t* to $t + 1$), $S(t)$ is given by $l(t + 1)/l(t)$, except for the last age which must be zero.
2. The top row of the Leslie matrix is not *m* but $m(t)S(t)$, often referred to as the fertility.
3. To create the Leslie matrix we first create a matrix that is one row and one column smaller than required and use the R function `diag` to assign the survivals. This matrix is then inserted into the required spaces of the Leslie matrix. An alternate method using a loop is also shown in the coding below.
4. The value of *r* is obtained by taking the log of λ. For reasons that are not clear if `log(Lambda)` is returned the R function `optimize` gives the following error message:

```
Error in optimize(RCALC, interval = c(1, 3), maximum = TRUE):
invalid function value in 'optimize'
```

However, `abs(log(Lambda))` does not produce this error, even though all values of `log(lambda)` are already positive (the same error message is generated if `Lambda` alone is returned.

5. As suggested in Chapter 2, the relationship of *r* to body size is plotted to check graphically that the optimum is more or less at the value given by `optimize`.

R CODE:

```
  rm(list=ls())                              # Clear workspace
  RCALC <- function(x) # Function to generate Lelsie matrix and
eigenvalue
{
  Maxage <- 50                               # Maximum age
  M       <- Maxage-   1                     # 1 less than the maximum
                                               age
  age    <- seq( from=1, to=Maxage)          # vector of ages
  Af     <- 0 ;Bf <- 16; As <-1 ; Bs <- 0.5  # Parameter values
  m      <- rep(Af+Bf*x, times=Maxage)       # number of female
                                               offspring
  l      <- exp(-(As+Bs*x)*age)              # Survival to age
  S      <- matrix(0,Maxage,1)               # Pre-assign space for
                                               age-specific survival
  S[1]   <- l[1]                             # Survival to age 1
# Calculate the survival from t to t+1
  for ( i in 2:M) {S[i] <- l[i]/l[i-1]}
  Fertility <- m*S                          # Top row of Leslie matrix
  Dummy     <- matrix(0,M,M)                # Create a temporary matrix
  diag(Dummy) <- S[1:M]            # Assign survivals to diagonal
  Leslie.matrix <- matrix(0, Maxage, Maxage) # Pre-assign space
  Leslie.matrix[1,] <- Fertility       # Add fertilities to top row
  Leslie.matrix[2:Maxage,1:M] <- Dummy # Add dummy to appropriate
                                         space
# An alternate approach using a loop is shown below
#j <- 0; for (i in 2:Maxage){j <- j+1 ;Leslie.matrix[i,j] <- S
[i-1]}
  Eigen.data <- eigen(Leslie.matrix)     # Call eigen
  Lambda <- Eigen.data$values[1]         # Get first eigenvalue
  return(abs(log(Lambda)))               # Return r
}
  Optimum <- optimize(RCALC, interval= c(1,3), maximum=TRUE)
  Best.X <- Optimum$maximum              # Optimum body size
  Best.r <- Optimum$objective            # Maximum r
# Print out results to 6 significant digits
  print(c('Best x = ', signif(Best.X, 6), 'Best r = ', signif
  (Best.r,6)))
  # Plot r.est vs x
  n      <- 50                                  # Nos of increments
  x      <- matrix(seq(from=1, to=2.5, length=n))  # Values of x
  r.est  <- apply(x,1,RCALC)                    # Get values of r
  plot(x, r.est,xlab="Body size, x", ylab="r.est", type='l')
  points( Best.X, Best.r, cex=2) # Add point to graph at optimum
```

OUTPUT:
Figure not shown but same as Figure 2.5, except for added point at optimum.

```
[1] "Best x = "  "1.93751"  "Best r = " "1.49699"
```

which agrees, as expected, with the previous results. The matrix approach is somewhat simpler in its coding compared to the summation approach of the last section but not to that using the explicit function that relates r to x (equation (3.25)).

3.4 Scenario 2: Adding density-dependence

We continue with the previous model but add density-dependence and use pairwise invasibility and elasticity analyses to locate the optimum body size.

3.4.1 General assumptions

1. All assumptions given in Scenario 1 hold.
2. Population size is limited by density-dependence.

3.4.2 Mathematical assumptions

1. All assumptions given in Scenario 1 hold.
2. Population size is controlled by a Ricker density-dependent function:

$$F_t^* = F_t \alpha e^{-\beta N_t} \tag{3.26}$$

where N_t is total population size, F_i is the density-independent component of fertility as defined by equation (3.20), and F_t* is the density-dependent fertility.

3.4.3 Solving using R_0 as the fitness measure

Because the density-dependence does not directly affect the trait under consideration the appropriate measure of fitness is not r but R_0. The relevant equation is equation (3.24) rewritten as

$$\frac{(a_F + b_F x)e^{-(a_S+b_S x)}}{1 - e^{-(a_S+b_S x)}} = R_0 \tag{3.27}$$

We can use the previous coding, modified for the change in function, to find the value of x that maximizes R_0 (alternatively, one could find x such that $\frac{dR_0}{dx} = 0$).

R CODE:

```
  rm(list=ls())                          # Clear workspace
  RCALC <- function(x)                   # Function to calculate r
{
  Af <-0; Bf<-16; As<-1; Bs<-0.5         # parameter values
  A <- Af+Bf*x; B <- As+Bs*x             # For convenience
  R0 <- A*exp(-B)/(1-exp(-B))            # R0
  return(R0)                             # return value
}
# Call optimize to find best x
  optimize(f=RCALC,interval= c(0.1,3), maximum=TRUE)$maximum
```

OUTPUT:

```
[1] 1.682795
```

Although the density-dependence does not directly involve body size it changes the operational fitness measure and hence also the optimal body size. We now examine the approaches of invasibility and elasticity analyses.

3.4.4 Pairwise invasibility analysis

The program follows the pattern outlined in the introduction with minor changes. A general description of the functions follows:

1. `LESLIE <- function(x,Maxage)`: This is the same function as in the previous scenario and constructs the Leslie matrix from the relevant equations.

2. `DD.FUNCTION <- function(ALPHA, BETA, Fi, n) {Fi*ALPHA*exp(-BETA*n)}`: This passes the two density dependent parameters, the DI (density-independent) fertility coefficient and total population size, and passes back the new fertility as defined by equation (3.26). Parameter values are set at $\alpha = 1$, $\beta = 2 \times 10^{-5}$, which produces a stable equilibrium of 80782.
3. `POP.DYNAMICS <- function(X)`: X contains the body size of the resident and the body size of invader (this differs from the example in the introduction which passed the multiplier for the invader). The function calculates the growth rate of the invader. Unlike the example given in the introduction the trajectory of the resident-only population is not followed.
4. Main program: This follows the approach outlined in the introduction. The body size obtained from the elasticity analysis that follows is plotted onto the contour surface.

R CODE:

```
  rm(list=ls()) # Clear workspace
  LESLIE <- function(x,Maxage) # Function to generate Leslie matrix
{
  M     <- Maxage-1                    # 1 less than the maximum age
  age   <- seq( from=1, to=Maxage)        # vector of ages
  Af    <- 0 ;Bf <- 16; As <-1 ; Bs <- 0.5   # Parameter values
  m     <- rep(Af+Bf*x, times=Maxage) # number of female offspring
  l <- exp(-(As+Bs*x)*age)             # Survival to age
  S <- matrix(0,Maxage,1)        # Space for age-specific survival
  S[1] <- l[1]                   # Survival to age 1
# Calculate the survival from t to t+1
  for ( i in 2:M) {S[i] <- l[i]/l[i-1]}
  Fertility       <- m*S              # Top row of Leslie matrix
  Dummy           <- matrix(0,M,M) # Create a temporary matrix
  diag(Dummy)     <- S[1:M]          # Assign survivals to diagonal
  Leslie.matrix   <- matrix(0, Maxage, Maxage)  # Pre-assign space
  Leslie.matrix[1,] <- Fertility       # Add fertilities to top row
  Leslie.matrix[2:Maxage,1:M] <- Dummy   # Add dummy to appropriate
                                            space
  return(Leslie.matrix)
} # End of Leslie function
############### Density-dependence function ###############
  DD.FUNCTION   <-   function(ALPHA,BETA,Fi,n)   {Fi*ALPHA*exp
  (-BETA*n)}
###############Population dynamics function ###############
  POP.DYNAMICS <- function(X)
  {
  X.Resident <- X[1]    # Body size of resident population
  X.invader  <- X[2]    # Body size of invader
  ALPHA  <- 1 ; BETA <- 2*10ˆ-5    # Density dependence parameters
  Maxage <- 50                     # Maximum age
  Resident.matrix <- LESLIE(X.Resident, Maxage) # Resident Leslie
                                                  matrix
  Invader.matrix  <- LESLIE(X.invader, Maxage) # Invader leslie
                                                  matrix
  F.resident <- Resident.matrix[1,]      # Resident DI fertility
  F.invader  <- Invader.matrix[1,]       # Invader DI fertility
  Maxgen     <- 30                       # Nos of gens to run
  n.resident <- matrix(0,Maxage,1) # Resident population vector
  n.resident[1] <- 1               # Initial resident popn size
  for ( Igen in 2:Maxgen)          # Iterate over generations
{
  N <- sum(n.resident)             # Total popn size
# Get DD fertility for resident population at time Igen
  Resident.matrix[1,] <- DD.FUNCTION(ALPHA, BETA, F.resident, N)
  n.resident   <- Resident.matrix%*%n.resident   # Resident popn
} # End of first Igen loop
### Introduce invader ####
  Maxgen <- 100                # Number of generations to run
# Pre-allocate space for storage of invader population numbers
  Pop.invader      <- matrix(0,Maxgen,1)
# Pre-allocate space for invader vector
  n.invader        <- matrix(0,Maxage,1)
  n.invader[1]     <- 1                # Initial number of invaders
  Pop.invader[1,1]   <- n.invader[1] # Store initial numbers of
                                       invaders
  for (Igen in 2:Maxgen)               # Iterate over generations
{
# Total number in population.
  N <- sum(n.resident) + sum(n.invader)
# DD fertility of resident
  Resident.matrix[1,] <- DD.FUNCTION(ALPHA, BETA, F.resident, N)
# New resident vector
  n.resident          <- Resident.matrix%*%n.resident
# DD fertility of invader
  Invader.matrix[1,]  <- DD.FUNCTION(ALPHA, BETA, F.invader, N)
# New invader vector
  n.invader           <- Invader.matrix%*%n.invader
  Pop.invader[Igen]   <- sum(n.invader) # Store invader popn size
} # End of second Igen loop
# Now do linear regression of log(Pop.invader) on Generation
  Generation  <- seq(from=1, to=Maxgen) # Generate generation vector
  Nstart      <- 20                      # Generations to ignore
# Linear regression
  Invasion.model <-
  lm(log(Pop.invader[Nstart:Maxgen])~Generation[Nstart:Maxgen])
```

```
# Elasticity value = regression slope
  Elasticity <- Invasion.model$coeff[2]
} # End of POP.DYNAMICS function
######################## MAIN PROGRAM #######################
  N1           <- 30 # Nos of increments
  X.Resident   <- seq(from=1, to= 3, length=N1) # Resident body sizes
  X.Invader    <- X.Resident                     # Invader body sizes
  d   <- expand.grid(X.Resident, X.Invader)  # Combinations
# Generate values at combinations
  z            <- apply(d,1,POP.DYNAMICS)
  z.matrix     <- matrix(z, N1, N1)         # Convert to a matrix
# Plot contours
contour(X.Resident,    X.Invader,z.matrix,    xlab="Resident",
ylab="Invader")
# Place circle at predicted optimal body size
  points( 1.68703, 1.68703, cex=3)    # cex triples size of circle
```

OUTPUT: (Figure 3.5)

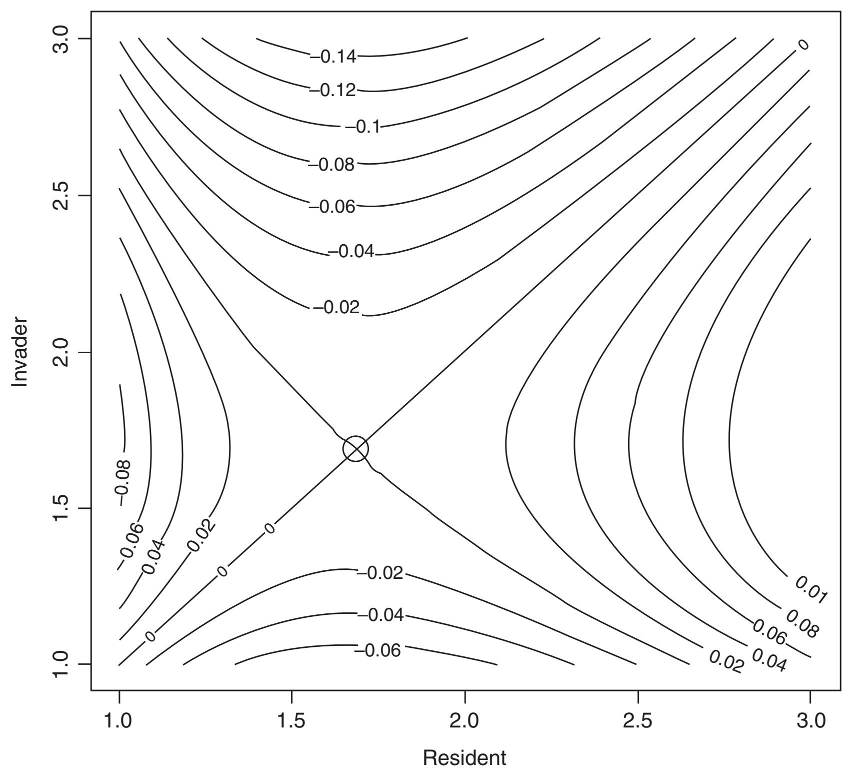

Figure 3.5 Pairwise invasibility plot for Scenario 2. The circle indicates the value obtained from the elasticity analysis.

There is a single putative ESS, which, as shown by the circle, is confirmed by the elasticity analysis described below.

3.4.5 Elasticity analysis

The program consists of the same components with the following changes

1. `POP.DYNAMICS(X, Coeff)`: The body size of the resident and the multiplier for the invader is passed to `POP.DYNAMICS`. As suggested by Benton and Grant (1999), the value of the invader trait is set at 0.995 times that of the resident (`Coeff = 0.995` in call to `POP.DYNAMICS` and hence `X.invader = 0.995*X.resident`). The change in population size of the invader population, `Pop.invader` is stored and after the specified number of generations (`Maxgen`) the elasticity is estimated as the slope of the linear regression of `log(Pop.invader)` on `Generation`. To avoid poor estimates due to the initial stabilization of the invader population the regression ignores the first 20 generations.
2. The optimum body size is that value at which elasticity is zero. This is estimated first by calling the R function `uniroot` and then visually checked by plotting elasticity versus body size and also the invasion exponent versus body size.

R CODE:

```
  rm(list=ls())                    # Clear workspace
  LESLIE <- function(x,Maxage)     # Function to generate Leslie
                                     matrix
{
CODING SAME AS IN INVASIBILITY ANALYSIS
} # End of Leslie function
############### Density-dependence function ###############
  DD.FUNCTION      <-   function(ALPHA,BETA,Fi,n)  {Fi*ALPHA*exp
  (-BETA*n)}
###############Population dynamics function ###############
  POP.DYNAMICS     <- function(X, Coeff)
{
  X.Resident    <- X              # Body size of resident population
  X.invader     <- X.Resident*Coeff         # Body size of invader
  REST OF CODE SAME AS IN INVASIBILITY ANALYSIS
} # End of POP.DYNAMICS function
#######################  MAIN PROGRAM  ######################
  par(mfrow=c(2,2))          # Divide graphics page into quarters
# Plot elasticity vs x
  N.int <- 20                                # Number of increments
  X <- matrix(seq(from=.5, to=3, length=N.int))   # Sequence of
                                                    body sizes
# Calculate elasticities for sequence
  Elasticity <- apply(X,1,POP.DYNAMICS,0.995)
  plot(X, Elasticity, type='l') # Plot elasticity as a function of X
  lines(c(.5,3), c(0,0))        # Add horizontal line at zero
# Calculate the optimum by calling uniroot
  Optimum  <- uniroot(POP.DYNAMICS, interval=c(0.5,3),0.995)
  Best.X   <- Optimum$root # Save optimum X
  print(c("Optimum body size =",signif(Best.X,6))) # Print out value
# Now plot Invasion exponent when resident is optimal
# Note that because of order in call cannot use apply here
# Convert the X sequence to coefficients for call to POP.DYNAMICS
  Coeff     <- X/Best.X
  Invasion.exponent <- matrix(0,N.int,1) # Pre-allocate space
# Loop through values of X comparing to Best.X
for (i in 1:N.int){Invasion.exponent[i] <- POP.DYNAMICS(Best.X,
Coeff[i]) }
  plot(X, Invasion.exponent, type='l') # Plot invasion exponent vs x
  points(Best.X, 0, cex=2)             # Plot point at predicted
                                         optimum
```

OUTPUT: (Figure 3.6)

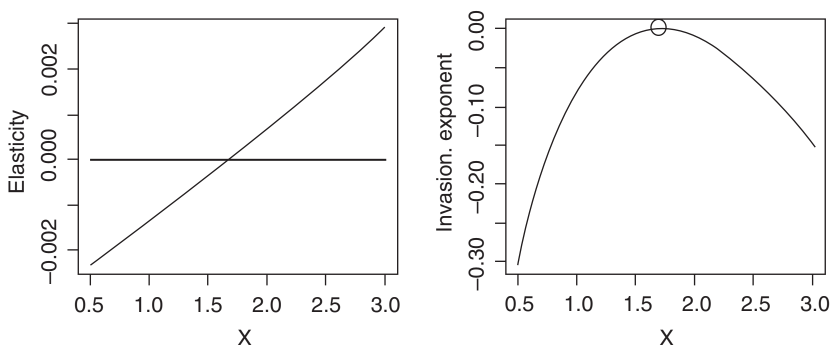

Figure 3.6 Graphical analysis of elasticity and invasion exponent as a function of body size as determined by an analysis of Scenario 2. The circle demarks the position of the predicted optimum.

```
[1] "Optimum body size =" "1.68703"
```

The results from `uniroot` indicate that the optimum under density-dependent regulation is smaller than in the density-independent case (Scenario 1). The plot of invasion exponent on body size confirms that the putative ESS is indeed an ESS.

3.5 Scenario 3: Functional dependence in the Ricker model

Ebenman et al. (1996) studied a stage-structured model in which selection favors stability, whereas oscillatory behavior is favored in the age-structure model studied by Greenman et al. (2005). In this model we consider how evolution will shape population dynamics by an analysis of the optimal parameter values in the Ricker model. No age or stage structure is assumed.

3.5.1 General assumptions

1. The organism is semelparous.
2. Recruitment is governed by a density-dependence function that allows for cyclical or chaotic population dynamics.
3. The parameters of the recruitment function are related such that the density-independent component is negatively related to the density-dependent component.

3.5.2 Mathematical assumptions

1. Population at time $t + 1$ is a Ricker function of the population at time t:

$$N_{t+1} = N_t \alpha e^{-\beta N_t} \tag{3.28}$$

2. The parameter α is a measure of density-independent recruitment whereas β is a measure of the density-dependent effect: increases in α increase recruitment but increases in β decrease recruitment by increasing the density-dependent component, $e^{-\beta N_t}$. Thus a positive functional relationship between α and β is indicative of a trade-off between the two recruitment components. For this scenario I shall assume the relationship

$$\beta = 0.001\alpha \tag{3.29}$$

Examples of the population dynamics for increasing values of α are shown in Figure 3.7. For low values of α the population reaches a stable equilibrium but as α is increased the dynamics first become cyclical and then chaotic.

3.5.3 Pairwise invasibility analysis

The coding follows the general pattern of that in Scenario 2, again plotting the value from the subsequent elasticity analysis on the contour plot. The `DD.FUNCTION` passes back the new population size using equation (3.28).

R CODE:

```
  rm(list=ls())                                # Clear memory
  DD.FUNCTION <- function(ALPHA,N1,N2)         # Density-dependence
                                                 function
{
  BETA <- ALPHA*0.001              # Set value of beta
  N <- N1*ALPHA*exp(-BETA*N2)      # New population size
  return(N)
} # End of DD.FUNCTION
```

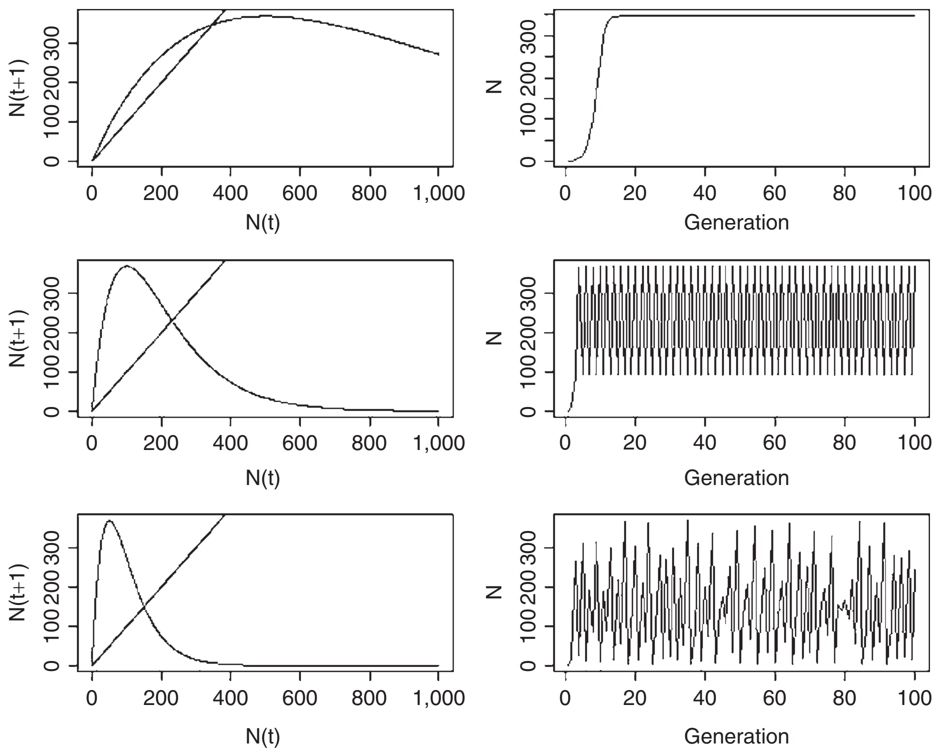

Figure 3.7 Population dynamics in the Ricker model in which $\beta = 0.001\alpha$. From top to bottom the values of α are 2, 10, and 20.

R CODING:

```
  rm(list=ls())        # Clear workspace
  par(mfrow=c(3,2))  # Divide graphics page into 3x2 panels
  DD.FUNCTION   <- function(n,ALPHA, BETA)  {ALPHA*exp(-BETA*n)}
# main program
  A   <- c(2,10,20)     # Values of alpha
  for( j in 1:3)        # Iterate over values of alpha
{
  N.t      <- seq(from=0, to=1000)       # Population sizes
  ALPHA    <-A[j]                        # alpha
  BETA     <- ALPHA*0.001                # Beta
# Plot N(t+1) vs N(t)
  N           <- length(N.t)     # Nos of values of N(t)
  N.tplus1   <- matrix(0,N)     # Pre-allocate space for N(t+1)
  for ( i in 1:N)               # Iterate over values of N
{
  N.tplus1[i]   <- N.t[i]*DD.FUNCTION(N.t[i],ALPHA,BETA)
}  # End of N(t+1) on N(t) calculation
  plot(N.t, N.tplus1, type='l', xlab='N(t)',ylab='N(t+1)')
  lines(N.t, N.t)                         # Plot the line of equality
# Plot N(t) vs t
  Maxgen      <- 100                      # Number of generations
  N           <- matrix(0, Maxgen)       # Pre-allocate space for N(t)
  N[1]        <- 1                        # Initial vale of N
  for   ( Igen in 2:Maxgen)               # Iterate over generations
{
  N[Igen]    <-  N[Igen-1]*DD.FUNCTION(N[Igen-1],ALPHA,BETA)
} # End of Igen loop
  Generation   <- seq(1,Maxgen)         # Vector of generation numbers
  plot(Generation, N, type='l')         # Plot population trajectory
} # End of j loop
```

```
#### Function specifying population dynamics ####
  POP.DYNAMICS <- function(ALPHA)
{
  ALPHA.resident <- ALPHA[1]      # Alpha for resident
  ALPHA.invader  <- ALPHA[2]      # Alpha for invader
  Maxgen1     <- 50               # Generations when only invader
                                    present
  Maxgen2     <- 300              # Generations after invader
                                    introduced
  Tot.Gen <- Maxgen1+Maxgen2      # Total number of generations
  N.resident <- N.invader <- matrix(0,Tot.Gen) # Allocate space
  N.resident[1]      <- 1         # Initial number of resident
  N.invader[Maxgen1]<- 1          # Initial number of invader
  for (Igen in 2:Maxgen1)         # Iterate over only resident
{
  N.resident[Igen]  <-  DD.FUNCTION(ALPHA.resident,  N.resident
[Igen-1], N.resident[Igen-1])
} # End of resident only period
# Now add invader
  J <- Maxgen1+1            # Staring generation of this period
  for (Igen in J:Tot.Gen)  # Iterate after introduction of invader
{
  N.total <- N.resident[Igen-1]+ N.invader[Igen-1]   # Total popn
                                                       size
# Resident population size
  N.resident[Igen]  <-  DD.FUNCTION(ALPHA.resident,  N.resident
[Igen-1],N.total)
# Invader population size
  N.invader[Igen] <- DD.FUNCTION(ALPHA.invader, N.invader[Igen-
1],N.total)
} # End of invasion period
  Generation <- seq(from=1, to=Tot.Gen) # Generation sequence
  Nstart <- 10 + Maxgen1            # Starting point for regression
# Regression model
  Invasion.model  <-  lm(log(N.invader[Nstart:Tot.Gen]) ̃ Genera-
tion[Nstart:Tot.Gen])
  Elasticity <- Invasion.model$coeff[2]    # Elasticity
return(Elasticity)
} # End of POP.DYNAMICS function
############# MAIN PROGRAM #############
  N1 <- 30 # Nos of increments
  A.Resident  <- seq(from=2, to= 4, length=N1)  # Resident alpha
  A.Invader   <- A.Resident                     # Invader alpha
  d      <- expand.grid(A.Resident, A.Invader) # Combinations
# Generate values at combinations
  z      <- apply(d,1,POP.DYNAMICS)
  z.matrix <- matrix(z, N1, N1)                # Convert to a matrix
# Plot contours
  contour(A.Resident, A.Invader,z.matrix, xlab="Resident",
  ylab="Invader")
# Place circle at predicted optimal body size
points(2.725109, 2.725109, cex=3)     # cex triples size of circle
```

OUTPUT: (Figure 3.8)

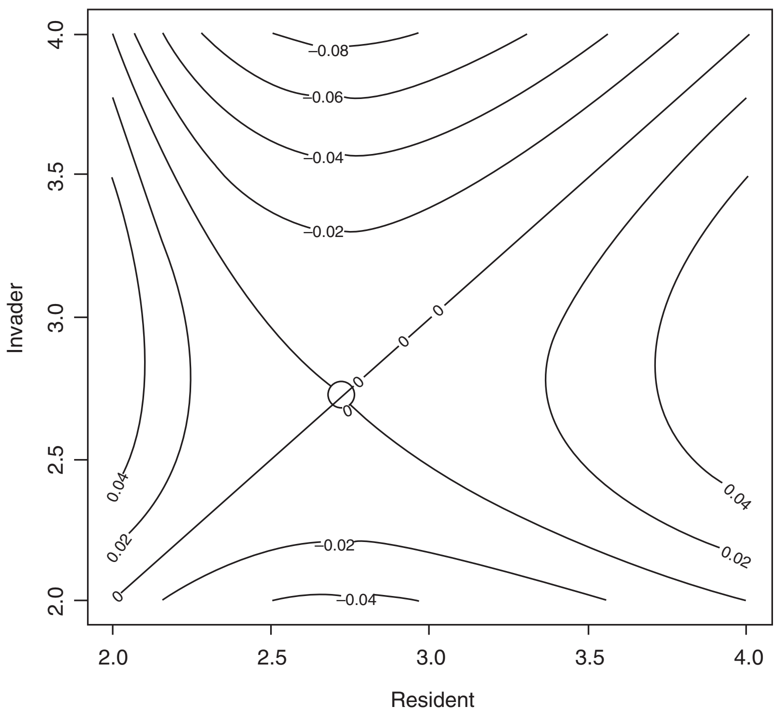

Figure 3.8 Paiwise invasibility plot for Scenario 3. The circle demarks the combination obtained from the elasticity analysis.

There is a single putative ESS which corresponds to the value obtained from the elasticity analysis described below.

3.5.4 Elasticity analysis

In addition to the two plots produced from the elasticity analysis two further plots are produced, $N(t+1)$ on $N(t)$ and $N(t)$ on t. These plots show the population dynamics as a function of the optimal value of α.

R CODE:

```
  DD.FUNCTION  <-  function(ALPHA,N1,N2)  #  Density-dependence
                                             function
{
  BETA <- ALPHA*0.001              # Set value of beta
  N <- N1*ALPHA*exp(-BETA*N2)      # New population size
  return(N)
} # End of DD.FUNCTION
#### Function specifying population dynamics ####
  POP.DYNAMICS <- function(ALPHA, Coeff)
{
  ALPHA.resident <- ALPHA                   # Alpha for resident
  ALPHA.invader <- ALPHA.resident*Coeff     # Alpha for invader
  REST OF CODING SAME AS IN INVASIBILITY ANALYSIS
} # End of POP.DYNAMICS function
###############   MAIN PROGRAM   ###############
  par(mfrow=c(2,2))           # Divide graphics page into quadrats
# Call uniroot to find optimum
  minA          <-1; maxA <-10 # Limits for search
  Optimum       <-     uniroot(POP.DYNAMICS,     interval=c(minA,
                                                 maxA),0.995)
  Best.Alpha    <- Optimum$root       # Store optimum Alpha
    print(Best.Alpha)                 # Print out optimum
# Plot Elasticity vs alpha
  N.int      <- 30 # Nos of intervals for plot
  Alpha      <-  matrix(seq(from=minA,  to=maxA,  length=N.int),
N.int,1)
  Elasticity <- apply(Alpha,1,POP.DYNAMICS, 0.995) # Get elastici-
                                                      ties
  plot(Alpha, Elasticity, type='l')
  lines(c(minA,maxA), c(0,0))   # Add horizontal line at zero
# Plot Invasion exponent when resident is optimal
  Coeff <- Alpha/Best.Alpha      # Convert alpha to coefficient
  Invasion.coeff <- matrix(0,N.int,1) # Allocate space
# Calculate invasion coefficient
  for  (i  in  1:N.int){  Invasion.coeff[i]  <-  POP.DYNAMICS(Best.
                                             Alpha, Coeff[i]) }
  plot(Alpha, Invasion.coeff, type='l')   # Plot invasion coeff on
                                           alpha
  points(Best.Alpha,0, cex=2)        # Plot optimum alpha on graph
# Plot N(t+1) on N(t) for optimum alpha
  maxN      <- 1000                   # Number of N
  N.t       <- seq(from=1, to=maxN)   # Values of N(t)
  N.tplus1 <- matrix(0,maxN)          # Allocate space for N(t+1)
  for ( i in 1:maxN)                  # Iterate over values of N
{
  N.tplus1[i] <- DD.FUNCTION(Best.Alpha, N.t[i], N.t[i])
}  # End of i loop
  plot(N.t, N.tplus1, type='l', xlab='N(t)', ylab='N(t+1)')
# Plot N(t) on t
  N <- matrix(1,100) # Allocate space. Note reuse of N
  for (i in 2:100){N[i]<- DD.FUNCTION(Best.Alpha, N[i-1], N[i-1])}
  plot(seq(from=1, to=100), N, type='l', xlab='Generation',
ylab='Population')
```

OUTPUT: (Figure 3.9)

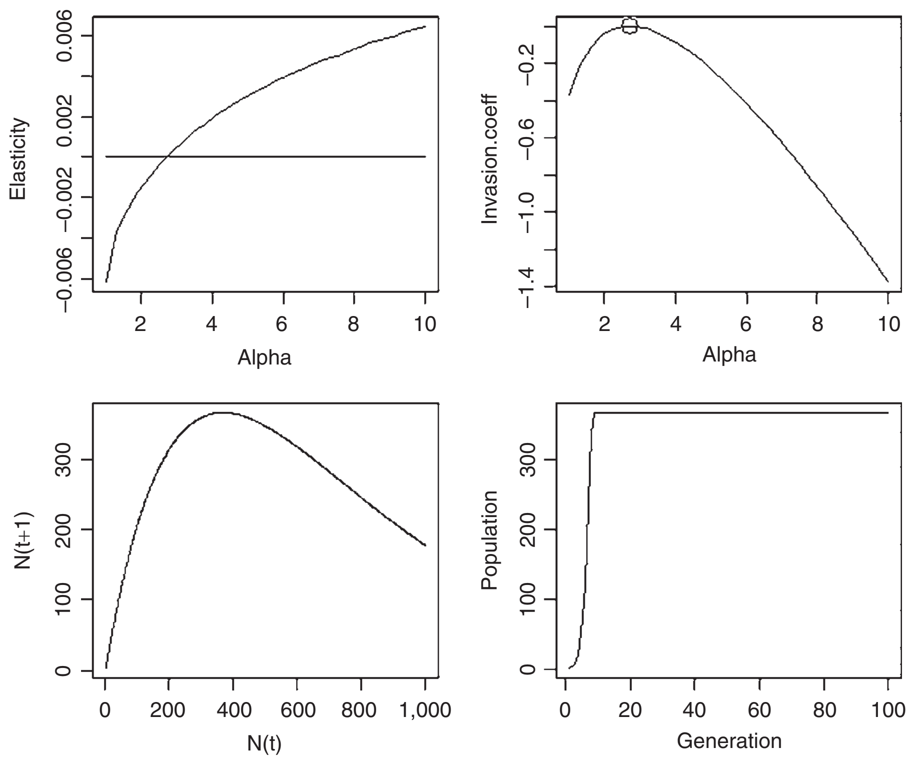

Figure 3.9 Output from the elasticity analysis of Scenario 3.

```
>  print(Best.Alpha) # Print out optimum
[1] 2.725107
```

At the optimal value of α the population reaches a stable equilibrium (Figure 3.9). Thus in this case selection favors stability.

3.5.5 Multiple invasibility analysis

The program follows the outline previously given and introduces a single random mutant in each generation. The simulation is run for 5,000 generations.

R CODE:

```
 rm(list=ls())                                 # Clear memory
  DD.FUNCTION<- function(X, N.total)           # Density-dependence
                                                 function
{
# Set parameter values
  ALPHA    <- X[1]                # Set alpha
  N        <- X[2]                # Population size for this alpha
  BETA     <-   ALPHA*0.001       # Set value of beta
  N    <- N*ALPHA*exp(-BETA*N.total)        # New cohort size
  return(N)
} # End of function
############## MAIN PROGRAM ##############
  set.seed(10)               # Initialize the random number seed
  Maxgen  <- 5000            # Number of generations run
  Stats    <- matrix(0,Maxgen,3) # Allocate space for statistics
  MaxAlpha    <- 4                  # maximum value of alpha
  Ninc        <- 50                 # Number of classes for alpha
# Allocate space to store data for each generation
  Store       <- matrix(0,Maxgen, Ninc)
# Allocate space for alpha class and population size
  Data        <- matrix(0,Ninc,2)
  Data[24,2] <- 1      # Initial population size and alpha class
  ALPHA   <-   matrix(seq(from=2,   to=MaxAlpha,   length=Ninc),
Ninc,1)  # Set Alpha
  Data[,1] <- ALPHA             # Place alpha in 1st column
  for (Igen in 1:Maxgen)        # Iterate over generations
{
  N.total   <- sum(Data[,2])              # Total population size
  Data[,2]  <- apply(Data,1,DD.FUNCTION, N.total)   # New cohort
  Store[Igen,]     <- Data[,2]   # Store values for this generation
# Keep track of population size, mean trait value and SD of trait
value
  Stats[Igen,2]    <- sum(Data[,1]*Data[,2])/sum(Data[,2])  # Mean
  S              <- sum(Data[,2])             # Population size
  Stats[Igen,1]      <- S                     # Population size
  SX1          <- sum(Data[,1]^2*Data[,2])
  SX2          <- (sum(Data[,1]*Data[,2]))^2/S
  Stats[Igen,3]     <- sqrt((SX1-SX2)/(S-1))   # SD of trait
```

```
# Introduce a mutant by picking a random integer between 1 and 50
  Mutant          <- ceiling(runif(1, min=0, max=50))
  Data[Mutant,2]  <- Data[Mutant,2]+1     # Add mutant to class
} # End of Igen loop
  par(mfrow=c(2,2))  # Split graphics page into quadrats
#   Plot last row of Store
  plot(ALPHA,    Store[Maxgen,],    type='l',    xlab='Alpha',
ylab='Number')
#   plot(Data.out[,1], Data.out[,2],
  Generation <- seq(from=1, to=Maxgen)
  N0         <- 1
  plot(Generation[N0:Maxgen],   Stats[N0:Maxgen,1],   ylab='Pop-
ulation size', type='l')
  plot(Generation[N0:Maxgen],  Stats[N0:Maxgen,2],  ylab='Mean',
type='l')
  plot(Generation[N0:Maxgen],   Stats[N0:Maxgen,3],   ylab='SD',
type='l')
  print(c('Mean alpha in last gen = ',Stats[Maxgen,2]))
  print(c('SD of alpha in last gen = ',Stats[Maxgen,3]))
```

OUTPUT: (Figure 3.10)

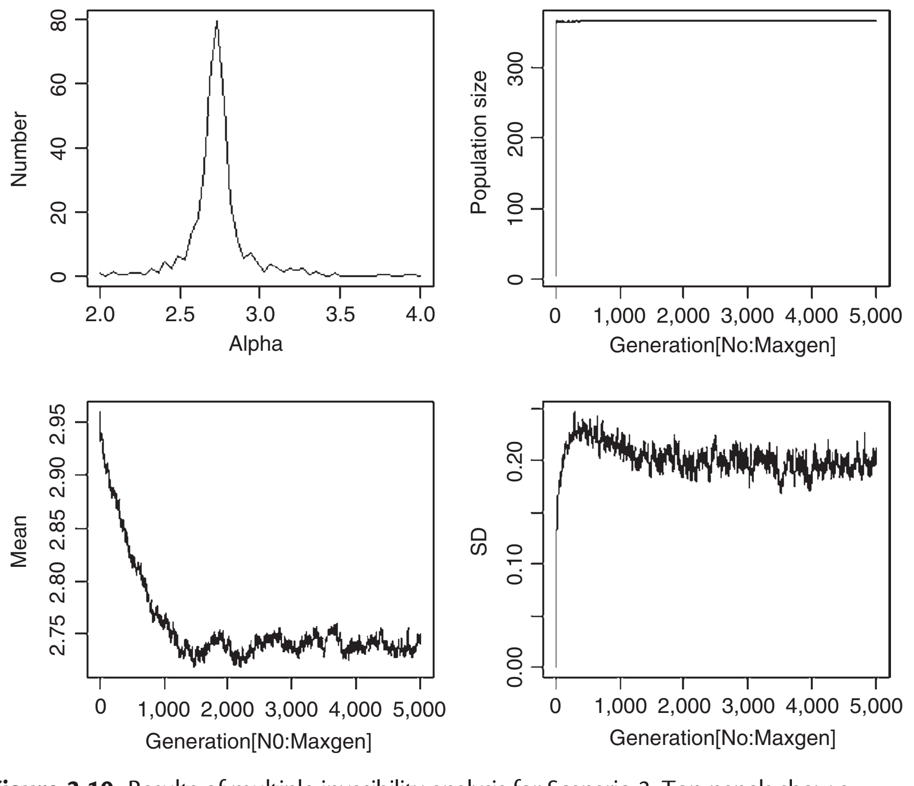

Figure 3.10 Results of multiple invasibility analysis for Scenario 3. Top panels show a frequency polygon of the distribution of α at the last generation and the population size over the simulation. The bottom row shows the change in the mean and standard deviation of α.

```
>  print(c('Mean alpha in last gen = ',Stats[Maxgen,2]))
[1]  "Mean alpha in last gen = " "2.74127304258184"
>  print(c('SD of alpha in last gen = ',Stats[Maxgen,3]))
[1]  "SD of alpha in last gen = " "0.198511654428470"
```

As indicated by the previous analyses, the population evolves to a single equilibrium. While there is variation about the mean value, it is not significantly different from that obtained from the elasticity analysis (2.74 in this analysis compared to 2.73 in the last).

3.6 Scenario 4: The evolution of reproductive effort

Complexity is added in this scenario by the addition of age structure and density-dependence that affects the parameter of interest, which here is reproductive effort. This scenario is taken from Benton and Grant (1999).

3.6.1 General assumptions

1. The population is composed of two age classes.
2. Fecundity increases with reproductive effort.
3. Survival decreases with reproductive effort.
4. Fecundity is a negative function of population size.

3.6.2 Mathematical assumptions

1. Fecundity, F_i, is defined as the number of offspring born to an individual that survive to the next age or stage.
2. Fecundity is the following function of reproductive effort, E, and population size:

$$F_i E \alpha e^{-\beta N} \tag{3.30}$$

where N is the total population size ($= n_1 + n_2$).

3. Survival is governed by the relationship

$$S_i(1 - E^Z) \tag{3.31}$$

The matrix model is thus

$$\begin{bmatrix} n_{1,t+1} \\ n_{2,t+1} \end{bmatrix} = \begin{bmatrix} F_1 E \alpha e^{-\beta N} & F_2 E \alpha e^{-\beta N} \\ S_1(1 - E^Z) & S_2(1 - E^Z) \end{bmatrix} \begin{bmatrix} n_{1,t} \\ n_{2,t} \end{bmatrix} \tag{3.32}$$

The value of Z was set at 6 which matches approximately and corresponds to the mean value found in lowland birds (Benton and Grant 1999). The Ricker parameters were set at $\alpha = 1$ and $\beta = 2 \times 10^{-5}$: these parameters produce a stable population.

3.6.3 Pairwise invasibility analysis

The function `DD.FUNCTION` calculates the density-dependent fertilities according to equation (3.30); otherwise the program follows the same pattern as in the previous two scenarios.

R CODE:

```
  rm(list=ls())   # Clear workspace
# Density-dependent function
  DD.FUNCTION <- function(ALPHA, BETA, F.DI, Ei, n)
  {Ei*F.DI*ALPHA*exp(-BETA*n)}
# Function to calculate dynamics of invasion
  POP.DYNAMICS <- function(E)
{
  E.resident   <- E[1]              # E for resident
  E.invader   <- E[2]               # E for invader
  F.DI     <- c(4, 10)              # DI Fertilities
  S.DI     <- c(0.6,0.85)           # DI Survivals
  ALPHA   <- 1; BETA <- 2*10^-5     # DD parameter values
  z       <- 6                      # RE survival parameter value
# Preallocate space for matrices
  Resident.matrix   <-matrix(0,2,2) # Pre-assign space for matrix
  Invader.matrix    <-matrix(0,2,2) # Pre-assign space for matrix
# Apply reproductive effort to F
  Resident.matrix[1,]  <- E.resident*F.DI
  Invader.matrix[1,]   <- E.invader*F.DI
# Apply reproductive effort to S
  Resident.matrix[2,]  <- (1-E.resident^z)*S.DI
  Invader.matrix[2,]   <- (1-E.invader^z)*S.DI
# Run Maxgen generation with resident only
  Maxgen          <- 20              # Nos of generations
  n.resident      <- c(1,0)          # Initial population vector
  for ( Igen in 2:Maxgen)          # Iterate over generations
{
# Calculate the new entries
  N              <- sum(n.resident)    # Pop size of residents
  Resident.matrix[1,]<- DD.FUNCTION(ALPHA, BETA, F.DI, E.resi-
dent, N)                                             # New Fs
  n.resident    <-Resident.matrix%*%n.resident    # New pop vector
} # End of first Igen loop
# Introduce invader
  Maxgen        <- 100                  # Set nos of generations to run
  Pop.invader   <- matrix(0,Maxgen,1)   # pre-allocate space of pop size
  Pop.invader[1,1]  <- 1                # Initial population size
  n.invader         <- c(1,0)           # Initiate invader pop vector
  for (Igen in 2:Maxgen)                # Iterate over generations
{
  N       <- sum(n.resident) + sum(n.invader)     # Total pop
# Apply density dependence to fertilities
  Resident.matrix[1,] <-DD.FUNCTION(ALPHA, BETA, F.DI, E.resident, N)
  Invader.matrix[1,]  <-DD.FUNCTION(ALPHA, BETA, F.DI, E.invader, N)
# Calculate new population vectors
  n.resident          <- Resident.matrix%*%n.resident
  n.invader           <- Invader.matrix%*%n.invader
  Pop.invader[Igen]   <- sum(n.invader)   # Store pop size of invader
} # End of second Igen loop
  Generation      <- seq(from=1, to=Maxgen)  # Create vector of
                                                generations
# Get growth of invader starting at generation 20
  Invasion.model  <-  lm(log(Pop.invader[20:Maxgen])~Generation
[20:Maxgen])
# Elasticity = slope of regression
  Elasticity      <- Invasion.model$coeff[2]
  return(Elasticity)
} # End of function
####################### MAIN PROGRAM #######################
par(mfrow=c(1,1))
N1           <- 30   # Nos of increments
E.Resident    <- seq(from=.2, to= .9, length=N1) # Resident body sizes
E.Invader     <- E.Resident                      # Invader body sizes
d     <- expand.grid(E.Resident, E.Invader)      # Combinations
# Generate values at combinations
z             <- apply(d,1,POP.DYNAMICS)
z.matrix      <- matrix(z, N1, N1)           # Convert to a matrix
# Plot contours
contour(E.Resident, E.Invader,z.matrix, xlab="Resident", ylab
="Invader")
# Place circle at predicted optimal body size
points( 0.5651338, 0.5651338, cex=3)   # cex triples size of circle
```

OUTPUT: (Figure 3.11)

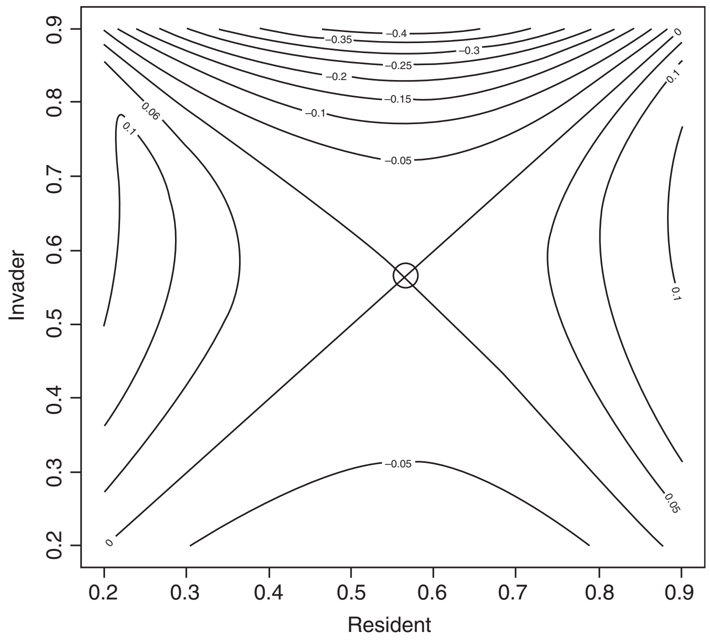

Figure 3.11 Pairwise invasibility plot for Scenario 4. The circle demarks the combination obtained from the elasticity analysis.

A single putative ESS is identified, corresponding to the predicted value from the following elasticity analysis.

3.6.4 Elasticity analysis

R CODE:

```
  rm(list=ls())   # Clear workspace
# Density-dependent function
  DD.FUNCTION   <- function(ALPHA, BETA, F.DI, Ei, n)
  {Ei*F.DI*ALPHA*exp(-BETA*n)}
# Function to calculate dynamics of invasion
  POP.DYNAMICS   <- function(E, Coeff)
{
  E.resident  <- E                # E for resident
  E.invader   <- E*Coeff          # E for invader
  REMAINDER OF CODING AS IN INVASIBILITY ANALYSIS
} # End of function
```

```
#################### MAIN PROGRAM ####################
  par(mfrow=c(2,2))   # Divide graphics page into quarters
# Locate value at which elasticity equals zero
  Optimum    <- uniroot(POP.DYNAMICS, interval=c(0.01,0.9),0.995)
  Best.E     <- Optimum$root # Store the optimum reproductive effort
  print(Best.E)              # Print optimum E
# Create plot of elasticity vs E
  N.int     <- 30            # Nos of increments
# Create sequence of E from .1 to 0.9 in N.int increments
  E      <- matrix(seq(from=0.2, to=.9, length=N.int), N.int,1)
# Create vector of elasticities using apply function
  Elasticity  <- apply(E,1,POP.DYNAMICS, 0.995)
  plot(E, Elasticity, type='l')        # Plot elasticity vs E
  lines(c(.2,.9), c(0,0))              # Add horizontal line at zero
# Now plot Invasion exponent when resident is optimal
  Coeff       <- E/Best.E      # Coefficient of invader DD function
  Invasion.exponent  <- matrix(0,N.int,1)  # pre-allocate space
# Iterate and calculate invasion exponent
# Note that a loop is used rather than apply because it is coeffi-
cient that
# is changing
  for (i in 1:N.int){Invasion.exponent[i]    <- POP.DYNAMICS(Best.
E, Coeff[i])
}
  plot(E, Invasion.exponent, ylab ='Invasion exponent', type='l')
  points(Best.E,0, cex=2) # Plot point at previously estimated
optimum E
```

OUTPUT: (Figure 3.12)

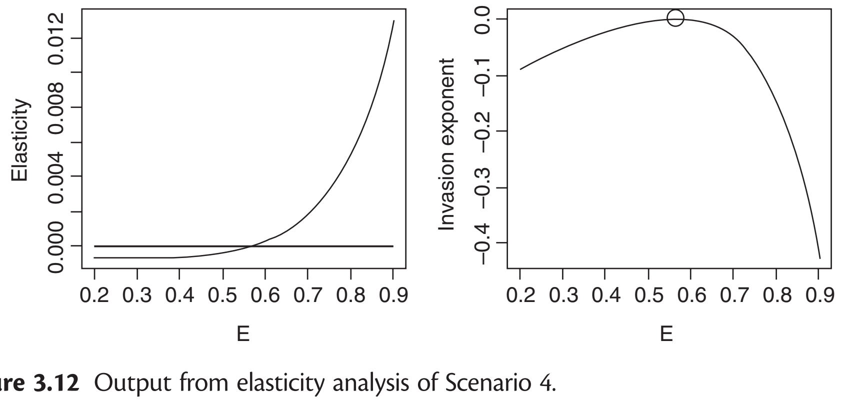

Figure 3.12 Output from elasticity analysis of Scenario 4.

```
> print(Best.E) # Print optimum E
[1] 0.5651338
```

The results indicate that under this scenario a medium reproductive effort is optimal. Benton and Grant (1999) explored a range of scenarios in which both the density-dependent function and traits affected were varied. The results are given in Table 1 of their paper: I suggest that the reader modify the above coding to replicate the results of that table.

3.7 Scenario 5: A two stage model

The primary purpose of this scenario is to illustrate the importance of correctly setting the number of generations over which the invader population growth rate is estimated. The problem is to estimate the optimal proportion of a population delaying maturity (or equivalently the proportion of immatures entering diapause or dormancy as found in many invertebrates and plants). For a detailed discussion of the model see van Dooren and Metz (1998). In this case I present the elasticity analysis first. Because the coding for multiple invasibility analysis is essentially the same as in previous scenarios, I have omitted it here.

3.7.1 General assumptions

1. The population consists of two stages with reproduction in the second stage.
2. A proportion of the first stage remains in that stage for more than one population cycle.
3. Fecundity is density dependent.

3.7.2 Mathematical assumptions

1. The proportion showing delayed maturity is P and the survival between stages is S.
2. Fecundity is determined by the Ricker function as $\alpha e^{-\beta A_t}$.

The matrix model is thus

$$\begin{bmatrix} I_{t+1} \\ A_{t+1} \end{bmatrix} = \begin{bmatrix} SP & \alpha e^{-\beta A_t} \\ S(1-P) & 0 \end{bmatrix} \begin{bmatrix} I_t \\ A_2 \end{bmatrix} \tag{3.33}$$

Parameter values are set at $S = 0.8$, $\alpha = 18$, and $\beta = 0.01$. The dynamics of the adult population depend critically on the proportion delaying maturity. Low values of P produce cyclical behavior and larger values produce a stable equilibrium, with the time attaining the equilibrium becoming shorter as P gets bigger (Figure 3.13). As a consequence the dynamics of the resident and invader populations, both may

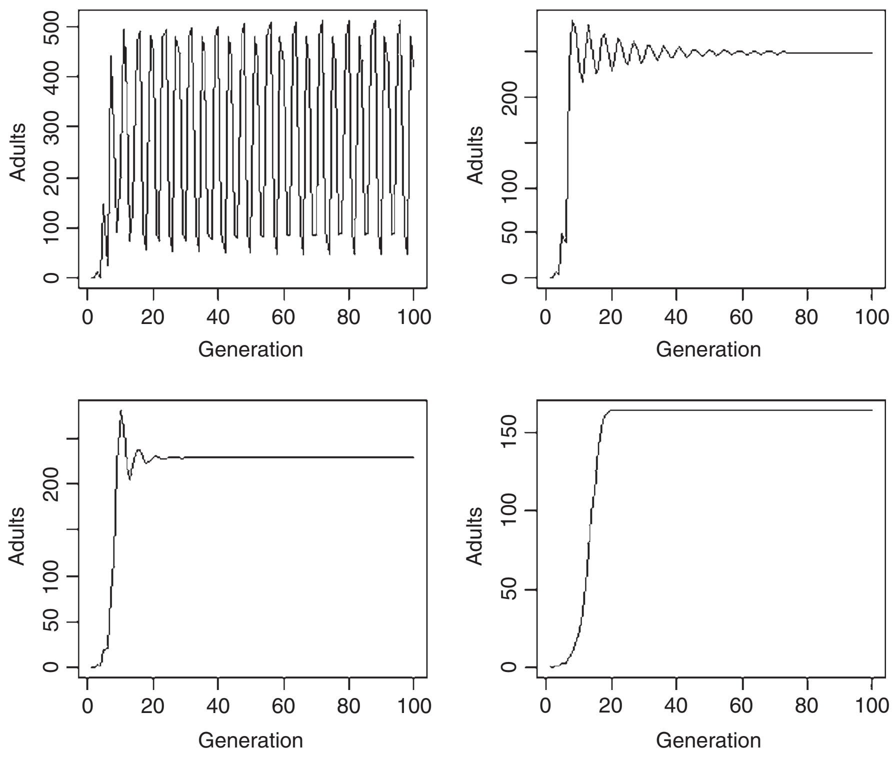

Figure 3.13 Population dynamics for the stage-dependent model of Scenario 5. Values of *P* from right to left and top to bottom are 0.1, 0.5, 0.7, and 0.9.

```
  rm(list=ls())           # Clear workspace
  par(mfrow=c(2,2))         # Divide graphics page into quarters
  ALPHA   <- 18 ;  BETA   <- 0.01;  S   <- 0.8 # Parameter values
  Ps      <- c(.1,.5,.7,.9)      #  Values of P
  for  ( Ith.P in 1:4)          # Iterate over P values
{
  P       <-  Ps[Ith.P]         # Select P value
  Stage   <- c(0,1)             # Start with one Adult
#  Initialize matrix
  Stage.matrix   <- matrix(c(S*P, S*(1-P), ALPHA*exp(-BETA*Stage
[2]),0),2,2)
  Maxgen        <- 100    # Nos of generations to run simulation
  Store.Stage<- matrix(0,Maxgen,2) # Preallocate space for storage
  Store.Stage[1,]   <- Stage    # Store generation 1
  for  (Igen in 2:Maxgen)      # Iterate over generations
{
  Stage               <- Stage.matrix%*%Stage      # New matrix
  Stage.matrix[1,2]  <- ALPHA*exp(-BETA*Stage[2]) # Apply DD func-
                                                    tion
  Store.Stage[Igen,]<- Stage                       # Store data
}  #  End of Igen loop
  plot(seq(from=1,  to=Maxgen),  Store.Stage[,2],xlab='Genera-
tion', ylab='Adults', type='l')
}  #  End of Ith.P loop
```

fluctuate, leading to uncertainty in the outcome of an invasion unless the time span is sufficiently long. Further, the variable dynamics may generate polymorphisms or multiple equilibria.

3.7.3 Elasticity analysis

Program coding follows the same pattern as in the previous scenario, except that the density dependence function is placed directly in the function POP.DYNAMICS. The line of coding shown in bold font is critical in this case.

R CODE:

```
  rm(list=ls())             # Clear workspace
  par(mfrow=c(1,1))         # Divide graphics page into quarters
  POP.DYNAMICS <- function(P, P.invader.coeff)
{
  P.resident      <- P                     # Resident delay
  P.invader       <- P*P.invader.coeff     # Invader delay
  ALPHA <- 18; BETA <- 0.01; S <- 0.8      # parameter values
  Resident.Stage  <- c(1,1)           # Initial resident population
  Invader.Stage   <- c(1,1)           # Initial invader population
# Initiate resident and invader matrices
  Resident.matrix     <-matrix(c(S*P.resident, S*(1-P.resident),
ALPHA*exp(-BETA*Resident.Stage[2]),0),2,2)
  Invader.matrix      <- matrix(c(S*P.invader, S*(1-P.invader),
ALPHA*exp(-BETA*Invader.Stage[2]),0),2,2)
  Maxgen              <- 100   # Generations with resident alone
  Store.Invader       <- matrix(0,Maxgen,2)
  Store.Invader[1,]   <- Invader.Stage
  for (Igen in 2:Maxgen)        # Iterate until resident is stable
{
  Resident.Stage   <- Resident.matrix%*%Resident.Stage   # New matrix
  Resident.matrix[1,2]  <- ALPHA*exp(-BETA*Resident.Stage[2])
# DD effect
} # End of 1st Igen loop
# Now enter invader
  Pop.invader    <- matrix(0,Maxgen,1)
  Maxgen         <- 100   # Set number of generations for invasion
  for (Igen in 1:Maxgen) # Iterate over generations after invasion
{
  N <- Resident.Stage[2]+ Invader.Stage[2]# Adult population size
  Resident.matrix[1,2]   <-ALPHA*exp(-BETA*N)  # Apply DD to resident
  Invader.matrix[1,2]    <-ALPHA*exp(-BETA*N)  # Apply DD to invader
# New matrices
  Resident.Stage     <- Resident.matrix%*%Resident.Stage
  Invader.Stage      <- Invader.matrix%*%Invader.Stage
  Pop.invader[Igen] <- Invader.Stage[2]  # Store invader adult pop
} # End of 2nd Igen loop
  Generation      <- seq(from=20, to=Maxgen)    # Generation vector
# Get invasion exponent from linear regression
  Invasion.model  <-  lm(log(Pop.invader[20:Maxgen])~Generation
[20:Maxgen])
  Elasticity      <- Invasion.model$coeff[2]
  } # End of POP.DYNAMICS
#################### MAIN PROGRAM ####################
  par(mfrow=c(2,2))        # Divide graphics page into 4 quadrats
# Call uniroot to find optimum
  Optimum    <- uniroot(POP.DYNAMICS, interval=c(0.1,0.6),0.995)
  Best.P     <- Optimum$root   # Store optimum P
  print(Best.P)                # Print out optimum
  N.int      <- 30             # Nos of intervals for plot
  P          <- matrix(seq(from=0.1, to=.6, length=N.int), N.int,1)
  Elasticity <- apply(P,1,POP.DYNAMICS, 0.995)
  plot(P, Elasticity, type='l')
  lines(c(.01,.9), c(0,0))      # Add horizontal line at zero
# Now plot Invasion exponent when resident is optimal
  Coeff           <- P/Best.P     # Convert P to coefficient
  Invasion.coeff <- matrix(0,N.int,1)
  for (i in 1:N.int){ Invasion.coeff[i] <- POP.DYNAMICS(Best.P,
Coeff[i])}
  plot(P, Invasion.coeff, type='l')
  points(Best.P,0, cex=2)
```

OUTPUT:

```
>  print(Best.P) # Print out optimum
[1] 0.3506968
```

The R function `uniroot` is successful in finding a root but it is quite clear from the graphical output (upper panels of Figure 3.14) that this is not the optimum. At first glance one might suppose that there are multiple ESS values, however, this variation disappears if the invader population is monitored for 2,000 generations and a single ESS value of 0.307579 is found (lower panels of Figure 3.14).

3.7.4 Pairwise invasibility analysis

Coding is omitted as it is essentially the same as in the previous two scenarios. What is important is that the plot generated after only 100 generations gives a remarkably accurate graphical estimate (Figure 3.15), as indicated by the superimposed plot from the elasticity analysis.

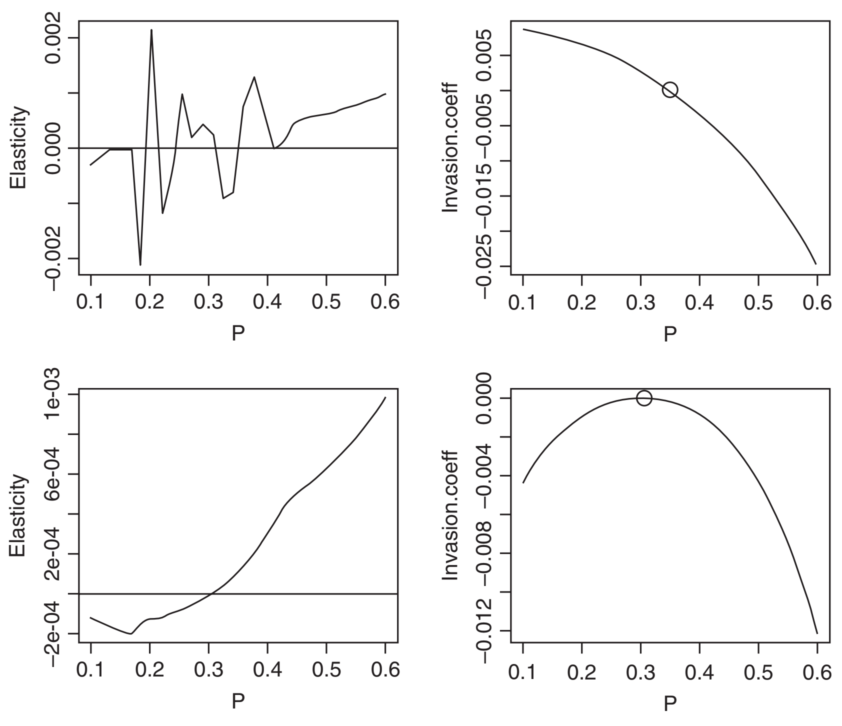

Figure 3.14 Elasticity analysis of Scenario 4. Top row shows results when invader is monitored for 100 generations and bottom row shows the results of monitoring for 2,000 generations.

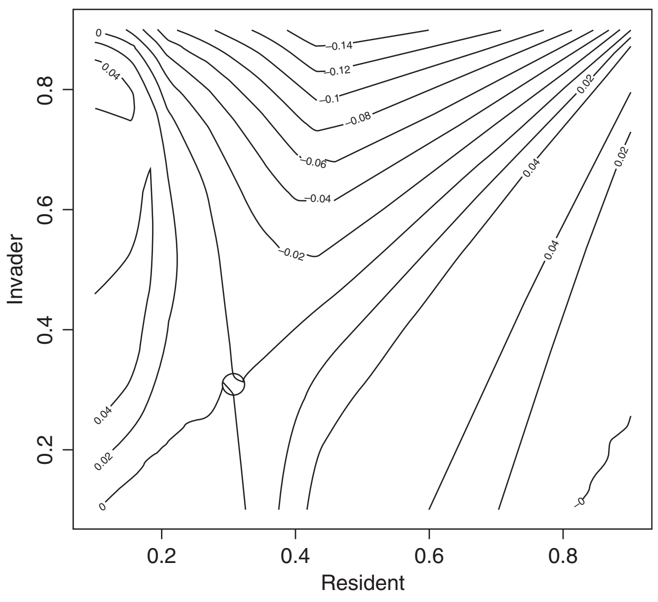

Figure 3.15 Pairwise invasibility plot of Scenario 5. The invading population was monitored for only 100 generations. Superimposed is the value (circle) predicted from the elasticity analysis run for 2,000 generations.

3.8 Scenario 6: A case in which the putative ESS is not stable

This scenario is discussed in detail in White et al. (2006). The present model is slightly modified to ensure that population size is reasonably large (>20). The results illustrate the importance of both an invasibility analysis and an elasticity analysis. The coding is essentially same as in previous scenarios.

3.8.1 General assumptions

1. Population size is governed by both density dependent and density-independent factors.
2. The density-independent components of fecundity and survival are negatively related.
3. The above function generates complex population dynamics.

3.8.2 Mathematical assumptions

1. Population size at time $t + 1$, N_{t+1}, is given by the equation

$$N_{t+1} = N_t(\alpha e^{-\beta N_t} + S_\alpha) \tag{3.34}$$

where S_α is survival as a function of α:

$$S_\alpha = \frac{(S_{\max} - S_{\min})\left(1 - \frac{\alpha-\alpha_{\min}}{\alpha_{\max}-\alpha_{\min}}\right)}{1 + a\frac{\alpha-\alpha_{\min}}{\alpha_{\max}-\alpha_{\min}}} \tag{3.35}$$

Depending on the value of α, S_α is a concave, convex, or linear function of α. In the present example parameter values are set as $S_{\min} = 0$, $S_{\max} = 0.9$, $\alpha_{\max} = 50$, $\alpha_{\min} = 1$, $a = 20$, and $\beta = 0.01$.

3.8.3 Pairwise invasibility analysis

R CODE:

```
  rm(list=ls())   # Clear memory
  DD.FUNCTION<-   function(ALPHA,N1,N2)  #   Density-dependence
function
{
# Set parameter values
  Amin <- 1; Amax <- 50; Bmin <- 0; Bmax <- 0.9; a <- 20; BETA <- 0.01
  AA   <- (ALPHA-Amin)/(Amax-Amin)       # For convenience
  S    <- (Bmax-Bmin)*(1-AA)/(1+a*AA)   # Survival
  N    <- N1*(ALPHA*exp(-BETA*N2)+S)    # new population
  return(N)
} # End of function
```

```
################ POP.DYNAMICS FUNCTION ################
POP.DYNAMICS <- function(ALPHA)
{
  ALPHA.resident <- ALPHA[1]              # Resident value
  ALPHA.invader  <- ALPHA[2]              # Invader value
  Maxgen1         <- 50         # Nos of generations for resident only
  Maxgen2         <- 300        # Nos of generations after invasion
  Tot.Gen         <- Maxgen1+Maxgen2    # Total number of generations
  N.resident <- N.invader <- matrix(0,Tot.Gen)    # Allocate space
  N.resident[1]      <- 1         # Initial pop size of resident
  N.invader[Maxgen1]<- 1          # Initial pop size of invader
  for (Igen in 2:Maxgen1)         # Iterate over generations with
                                    resident only
{
  N <- N.resident[Igen-1]
  N.resident[Igen] <- DD.FUNCTION(ALPHA.resident, N, N)
} # End of 1st Igen loop
# Now add invader
  J <- Maxgen1+1              # Starting generation
  for (Igen in J:Tot.Gen)     # Iterate over generations with invader
{
  N <- N.resident[Igen-1]+ N.invader[Igen-1]   # Total pop size
  N.resident[Igen] <-  DD.FUNCTION(ALPHA.resident,  N.resident
[Igen-1], N)
  N.invader[Igen]    <- DD.FUNCTION(ALPHA.invader, N.invader[Igen-
1], N)
} # End of 2nd Igen loop
  Generation <- seq(from=1, to=Tot.Gen) # Vector of generation
                                          numbers
  N0      <- 10+Maxgen1   # Gen at which to start regression analysis
  Invasion.model <- lm(log(N.invader[N0:Tot.Gen])~ Generation
[N0:Tot.Gen])
  Elasticity <- Invasion.model$coeff[2] # Elasticity coefficient
  return(Elasticity)
} # End of function
#################### MAIN PROGRAM ####################
  N1          <- 30 # Nos of increments
  A.Resident <- seq(from=5, to= 45, length=N1)  # Resident alphas
  A.Invader  <- A.Resident                      # Invader alphas
  d      <- expand.grid(A.Resident, A.Invader)  # Combinations
# Generate values at combinations
  z      <- apply(d,1,POP.DYNAMICS)
  z.matrix     <- matrix(z, N1, N1)            # Convert to a matrix
# Plot contours
```

```
contour(A.Resident,   A.Invader,   z.matrix,   xlab="Resident",
ylab="Invader")
# Place circle at predicted optimal alpha
points(24.21837, 24.21837, cex=3)   # cex triples size of circle
```

OUTPUT: (Figure 3.16)

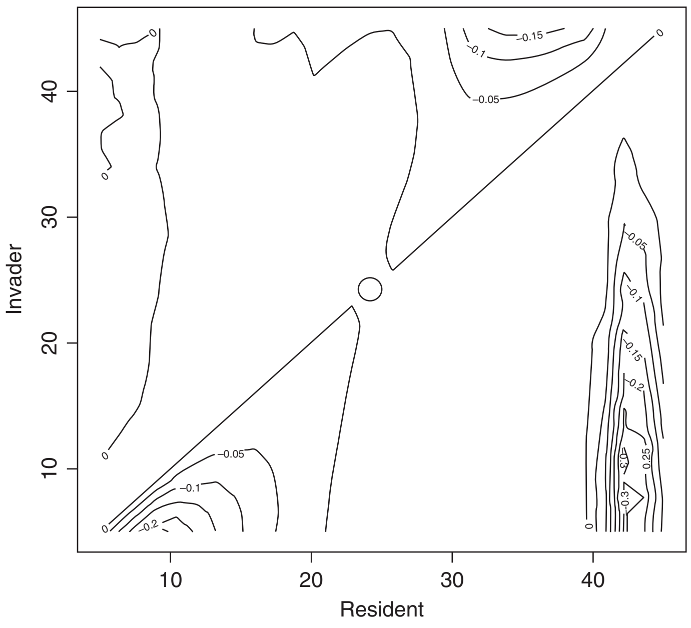

Figure 3.16 Pairwise invasibility plot for Scenario 6. The circle indicates the optimum obtained from the elasticity analysis.

There is a single putative ESS value that is at the point predicted by the elasticity analysis given below but the surface appears complex, with zero isoclines that do not intersect the $x = y$ line. Further, the one that does intersect the $x = y$ line is less than 90°, suggesting that the putative ESS is not stable (compare with the right-hand plots of Figure 3.3).

3.8.4 Elasticity analysis

R CODE:

```
  rm(list=ls())                               # Clear memory
  DD.FUNCTION <- function(ALPHA,N1,N2)   # Density-dependence
function
{
SAME AS IN INVASIBILITY ANALYSIS
} # End of function
```

```
################ POP.DYNAMICS FUNCTION ################
POP.DYNAMICS <- function(ALPHA, Coeff)
{
  ALPHA.resident   <- ALPHA          # Resident value
  ALPHA.invader    <- ALPHA*Coeff    # Invader value
  REMAINDER THE SAME AS IN INVASIBILITY ANALYSIS
} # End of function
################### MAIN PROGRAM ###################
  par(mfrow=c(2,2))      # Divide graphics page into quadrats
# Call uniroot to find optimum
  minA          <-10; maxA <-40       # Limits of search for alpha
  Optimum       <-     uniroot(POP.DYNAMICS,     interval=c(minA,
maxA),0.995)
  Best.A        <- Optimum$root       # Store optimum Alpha
  print(Best.A)                       # Print out optimum
  N.int         <- 30                 # Nos of intervals for plot
  A    <-matrix(seq(from=minA, to=maxA, length=N.int), N.int,1)
  Elasticity   <- apply(A,1,POP.DYNAMICS, 0.995)
  plot(A, Elasticity, type='l')
  lines(c(minA, maxA), c(0,0))          # Add horizontal line at zero
# Now plot Invasion exponent when resident is optimal
  Coeff             <- A/Best.A             # Convert A to coefficient
  Invasion.coeff <- matrix(0,N.int,1) # Allocate space
# Calculate invasion coefficient
  for (i in 1:N.int){ Invasion.coeff[i] <- POP.DYNAMICS(Best.A,
Coeff[i])}
  plot(A, Invasion.coeff, type='l')  # Plot invasion coeff vs alpha
  points(Best.A,0, cex=2)            # Add predicted optimum
# Plot N(t+1) on N(t) for best model
  N <- 1000                        # Nos of data points
  N.t <- seq(from=1, to=N)         # Values of N(t)
  N.tplus1 <- matrix(0,N)          # Allocate space
# Iterate to get N(t+1) on N(t)
  for ( i in 1:N){N.tplus1[i] <- DD.FUNCTION(Best.A, N.t[i], N.t
[i])}
  plot(N.t, N.tplus1, type='l')     # Plot N(t+1) on N(t)
# Plot N(t) on t
  N      <- matrix(0,100)            # Allocate space
  N[1]   <- 1                        # Initial value of N(t)
  for (i in 2:100){N[i]<- DD.FUNCTION(Best.A, N[i-1], N[i-1])}
  plot(seq(from=1,  to=100),  N,  type='l',  xlab='Generation',
ylab='Population')
```

OUTPUT: (Figure 3.17)

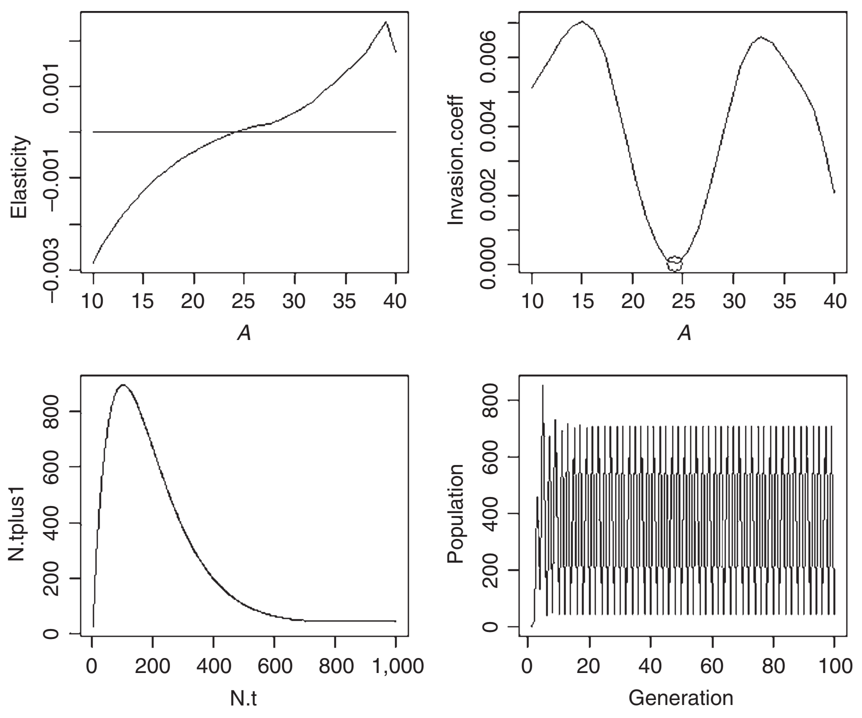

Figure 3.17 Elasticity analysis of Scenario 6.

```
>  print(Best.A) # Print out optimum
[1] 24.21837
```

The putative ESS is consistent with the pairwise invasibility analysis but the elasticity analysis does not indicate that the invasion coefficient is negative for values different from the putative ESS. In fact, the elasticity analysis suggests that invasion is highly likely. To view the population dynamics we can simply insert plot commands into the function `POP.DYNAMICS` (shown in bold font) and call this for the optimum α and a value for an invader.

R CODE:

```
rm(list=ls())                                 # Clear memory
  DD.FUNCTION  <- function(ALPHA,N1,N2)    # Density-dependence
function
{
SAME AS IN PREVIOUS
} # End of function
################ POP.DYNAMICS FUNCTION ################
POP.DYNAMICS <- function(ALPHA, Coeff)
{
  ALPHA.resident    <- ALPHA          # Resident value
  ALPHA.invader     <- ALPHA*Coeff    # Invader value
  Maxgen1          <- 50   # Nos of generations for resident only
  Maxgen2          <- 300      # Nos of generations after invasion
  Tot.Gen <- Maxgen1+Maxgen2   # Total number of generations
  N.resident <- N.invader <- matrix(0,Tot.Gen)   # Allocate space
  N.resident[1]       <- 1          # Initial pop size of resident
  N.invader[Maxgen1] <- 1          # Initial pop size of invader
  for (Igen in 2:Maxgen1)  # Iterate over generations with resident
                             only
{
  N <- N.resident[Igen-1]
  N.resident[Igen] <- DD.FUNCTION(ALPHA.resident, N, N)
} # End of 1st Igen loop
# Now add invader
  J <- Maxgen1+1            # Starting generation
  for (Igen in J:Tot.Gen)   # Iterate over generations with invader
{
  N <- N.resident[Igen-1]+ N.invader[Igen-1]   # Total pop size
  N.resident[Igen]  <-  DD.FUNCTION(ALPHA.resident,  N.resident
[Igen-1], N)
  N.invader[Igen]  <-  DD.FUNCTION(ALPHA.invader,  N.invader[Igen-
1], N)
} # End of 2nd Igen loop
  Generation  <-  seq(from=1,  to=Tot.Gen)  #  Vector  of  generation
                                              numbers
  plot (Generation, N.resident, xlab='Generation', ylab='Resident
N', type='l')
  plot (Generation, N.invader, xlab='Generation', ylab='Invader
N', type='l')
} # End of function
############## MAIN PROGRAM ##############
  par(mfrow=c(2,2))         # Divide graphics page into quadrats
  Best.A     <- 24.21635     # Best alpha from elasticity analysis
  Invader.A  <- 10           # Alpha for invader
  Coeff      <- Invader.A/Best.A  # Calculate relevant coefficient
  POP.DYNAMICS( Best.A,Coeff)     # Call functiom
```

OUTPUT: (Figure 3.18)

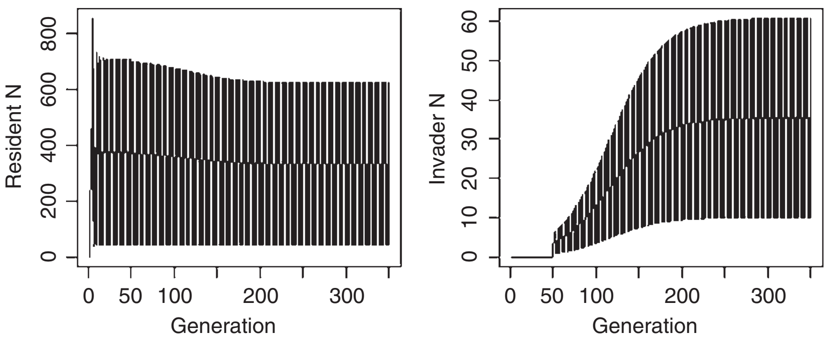

Figure 3.18 Temporal plots of resident and invader populations. Left-hand plot shows the population trajectory for the optimum α and the right-hand plot shows the population trajectory for an invader with $\alpha = 10$.

It is evident from the output that an invader with $\alpha = 10$ can invade the population but not replace the resident population. This explains the relation between the invasion coefficient and the optimum α and emphasizes the importance of elasticity analysis in conjunction with pairwise invasibility analysis.

3.8.5 Multiple invasibility analysis

The program is the same as in Scenario 6 except for: (*a*) it is run 10,000 generations to ensure sufficient time for equilibrium to be attained (if possible), (*b*) because the elasticity analysis indicated that a single equilibrium is unlikely, frequency polygons are plotted for the last three generations.

R CODE:

```
rm(list=ls())   # Clear memory
  DD.FUNCTION<- function(X, N.total)   # Density-dependence function
{
# Set parameter values
  ALPHA   <- X[1];     N <- X[2]
  Amin    <- 1; Amax <- 50; Bmin <- 0; Bmax <- 0.9; a <- 20; BETA <- 0.01
  AA      <- (ALPHA-Amin)/(Amax-Amin)          # For convenience
  S       <- (Bmax-Bmin)*(1-AA)/(1+a*AA)       # Survival
  N       <- N*(ALPHA*exp(-BETA*N.total)+S)  # new population
  N       <- max(0,N)                        # N cannot be negative
  return(N)
} # End of function
############## MAIN PROGRAM ##############
  set.seed(10)                 # Initialize the random number seed
  Maxgen   <- 10000            # Number of generations run
  Stats    <- matrix(0,Maxgen,3)  # Allocate space for statistics
  MaxAlpha <- 50                  # maximum value of alpha
  Ninc     <- 50                  # Number of classes for alpha
# Allocate space to store data for each generation
  Store       <- matrix(0,Maxgen, Ninc)
# Allocate space for alpha class and population size
  Data        <- matrix(0,Ninc,2)
  Data[24,2]  <- 1        # Initial population size and alpha class
  ALPHA <- matrix(seq(from=1, to=MaxAlpha, length=Ninc),Ninc,1)
# Set Alpha
  Data[,1] <- ALPHA            # Place alpha in 1st column
  for (Igen in 1:Maxgen)       # Iterate over generations
{
  N.total    <- sum(Data[,2])   # Total population size
  Data[,2]   <- apply(Data,1,DD.FUNCTION, N.total)   # New cohort
  Store[Igen,]   <- Data[,2]    # Store values for this generation
# Keep track of population size, mean trait value and SD of trait value
  Stats[Igen,2]  <- sum(Data[,1]*Data[,2])/sum(Data[,2])  # Mean
  S              <- sum(Data[,2])            # Population size
  Stats[Igen,1]  <- S                        # Population size
  SX1            <- sum(Data[,1]^2*Data[,2])
  SX2            <- (sum(Data[,1]*Data[,2]))^2/S
  Stats[Igen,3]  <- sqrt((SX1-SX2)/(S-1))    # SD of trait
# Introduce a mutant by picking a random integer between 1 and 50
  Mutant         <- ceiling(runif(1, min=0, max=50))
  Data[Mutant,2] <- Data[Mutant,2]+1       # Add mutant to class
} # End of Igen loop
  par(mfrow=c(3,2))       # Split graphics page into 3 x 2 panels
  for (Row in 9998:Maxgen) # Select rows to be plotted
{
  plot(ALPHA, Store[Row,], type='l', xlab='Alpha', ylab='Number')
} # End of frequency polygon plots
  Generation  <- seq(from=1, to=Maxgen)  # Vector of generations
  N0          <- 9900                    # Starting value for plots
  plot(Generation[N0:Maxgen], Stats[N0:Maxgen,1], ylab='Popula-
tion size',xlab='Generation', type='l')
  plot(Generation[N0:Maxgen], Stats[N0:Maxgen,2], ylab='Mean',
xlab='Generation', type='l')
  plot(Generation[N0:Maxgen],  Stats[N0:Maxgen,3],  ylab='SD',
xlab='Generation', type='l')
  print(c('Mean alpha in last gen = ',Stats[Maxgen,2]))
  print(c('SD of alpha in last gen = ',Stats[Maxgen,3]))
```

OUTPUT: (Figure 3.19)

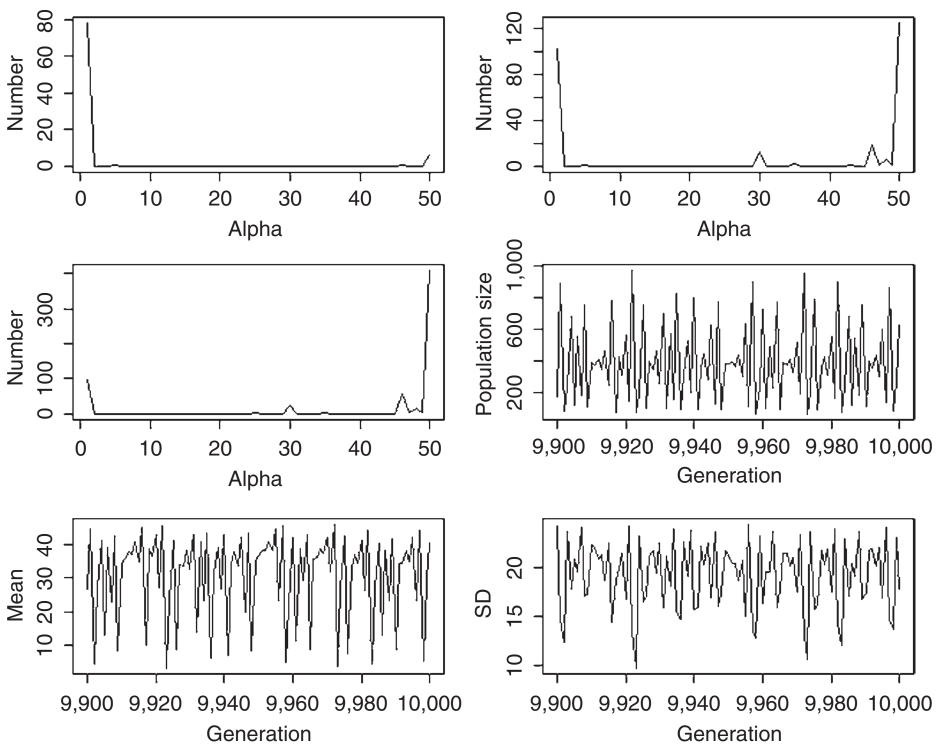

Figure 3.19 Results of multiple invasibility analysis of Scenario 6. The three frequency polygons show results for generation 9,998, 9,999 and 10,000.

```
>  print(c('Mean alpha in last gen = ',Stats[Maxgen,2]))
[1] "Mean alpha in last gen = " "40.6067676012847"
>  print(c('SD of alpha in last gen = ',Stats[Maxgen,3]))
[1] "SD of alpha in last gen = " "17.8941418836399"
```

The frequency polygon plots for the last three generations show dramatic shifts in the mean and modal values of α. As indicated by the previous analyses, the equilibrium from the pairwise invasibility analysis is highly unstable and is unlikely to be ever attained, since any small deviations set the population into extreme fluctuations. The population consists of different clones, with their frequencies changing through time.

3.9 Some exemplary papers

Katsukawa, Y., T. Katsukawa, and H. Matsuda. 2002. Indeterminate growth is selected by a trade-off between high fecundity and risk avoidance in stochastic environments. *Population Ecology* 44:265–272.

Problem: Optimal allocation of energy to growth, survival, and reproduction when survival is temporally variable

Model type: Two age/stage

Density-dependence: None

Trade-off: Total allocation of energy to survival, growth, and reproduction fixed

Analysis: Maximization of λ

Reference scenario: Scenario 1

Ebenman, B., A. Johansson, T. Jonsson, and U. Wennergren. 1996. Evolution of stable population dynamics through natural selection. Proceedings of the Royal Society of London – Series B: *Biological Sciences* 263:1145–1151.

Problem: Optimal combination of density-dependent parameters when population dynamics can range from a fixed equilibrium to chaotic

Model type: Two-stage

Density-dependence: Ricker function applied to maturation rate and juvenile survival

Trade-off: Maturation rate and juvenile survival

Analysis: Invasibility analysis

Reference scenario: Scenarios 3 and 5

Lalonde, R. G. and B. D. Roitberg. 2006. Chaotic dynamics can select for long-term dormancy. *American Naturalist* 168:127–131.

Problem: The optimal dormancy when population dynamics are chaotic

Model type: Up to 13 generations of dormancy

Density-dependence: Ricker function

Trade-off: No explicit trade-off

Analysis: Multiple invasibility analysis

Reference scenario: Scenarios 3 and 6

Johst, K., M. Doebeli, and R. Brandl. 1999. Evolution of complex dynamics in spatially structured populations. Proceedings of the Royal Society of London – Series B: *Biological Sciences* 266:1147–1154.

Problem: The optimal density-dependent and density-independent dispersal rates

Model type: One generation per time step

Density-dependence: Fecundity varies with density using an equation that can take a variety of shapes

Trade-off: No explicit trade-offs

Analysis: Multiple invasibility analysis

Reference scenario: Scenarios 3 and 6

Benton, R. A. and A. Grant. 1999. Optimal reproductive effort in stochastic, density-dependent environments. *Evolution* 53:677–688.

Problem: Optimal reproductive effort

Model type: Two age/stage class

Density-dependence: Various

Trade-off: Various combinations of reproductive effort and survival at age with or without temporal variation

Analysis: Elasticity analysis

Reference scenario: Scenario 4

Wilbur, H. M. and V. H. W. Rudolf. 2006. Life-history evolution in uncertain environments: Bet hedging in time. *American Naturalist* 168:398–411.

Problem: Optimal iteroparity and delayed maturity in a stochastic environment

Model type: Two-stage

Density-dependence: Ricker function applied to fertility with parameters selected to ensure a stable population in a deterministic environment

Trade-off: Adult survival and clutch size in some model variants

Analysis: Invasibility analysis

Reference scenario: Scenarios 4 and 5

White, A., J. V. Greenman, T. G. Benton, and M. Boots. 2006. Evolutionary behaviour in ecological systems with trade-offs and non-equilibrium population dynamics. *Evolutionary Ecology Research* 8:387–398.

Problem: Optimal reproductive effort when population dynamics are complex

Model type: No explicit age structure

Density-dependence: Ricker function

Trade-off: Density-independent fecundity and survival

Analysis: Invasibility analysis

Reference scenario: Scenario 6