Appendices
Advancing population ecology with integral projection models: a practical guide

Cory Merow, Johan P. Dahlgren, C.J.E. Metcalf,
Dylan Z. Childs, M.E.K. Evans, Eelke Jongejans, Sydne Record,
Mark Rees, Roberto Salguero-Gomez & Sean M. McMahon

Contents

Chapter 1

Appendix A: A simple IPM for a long-lived perennial plant

Contact: Cory Merow (cory.merow@gmail.com)

Abstract

Integral Projection Models (IPMs) use vital rate regressions to parameterize transition kernels that can be used for population projections. Here, we introduce an extremely simple IPM for a the long-live alpine perennial plant *Dracocephalum austriacum*. The analyses minimize the complexity of the R code in order to make the model transparent. After building a basic model, we illustrate ways to improve it, including, comparing alternative growth models, non-constant variance in growth, including a model for flowering probability. We calculate basic population statistics, including population growth rate, sensitivity, elasticity, and passage times throughout to check the plausibility of the model. We also illustrate a series of diagnostics, including correcting for eviction, determining asymptotic maximum size, and exploring the importance of transient dynamics. The model is associated with Example 1, Section II.A. in the main text.

1.1 Introduction

In this appendix, we illustrate how to build an IPM 'from scratch.' We begin with a simulated data set, build vital rate regressions, combine these regressions into an IPM kernel, and perform basic analyses of asymptotic dynamics. We begin with the simplest model of a perennial plant (corresponding to Example 1 in the main text) that one might consider for an IPM and iteratively improve the model to simulate the process by which a researcher might develop an IPM.

1.2 Study system/objectives

The goal of this exercise is to illustrate how to perform basic IPM analyses of asymptotic population growth rate, stable stage structure, reproductive values, sensitivities and elasticities of population growth rate to matrix elements. We then develop improvements to the model and illustrate the various diagnostics discussed in the main text to illustrate the iterative process of moving between data, models, and predictions to produce a biologically robust IPM. The data are simulated to clearly illustrate some of the subtle patterns we extract with the models, however the simulations derive from a model for a single annual transition (2001-2002) of the alpine plant *Dracocephalum austriacum*. Nicole et al. (2011) describe *Dracocephalum austriacum* as follows (see references therein):

'The Austrian Dragonhead (D. austriacum L., Lamiaceae) is a long-lived perennial plant occurring from the Pyrenees to the Caucasus, with scattered populations in Western Europe (Bensettiti et al. 2002). The plant is 20 to 50 cm high, with hairy stems and large blueâĂŞviolet insect-pollinated flowers (3.5 to 5 cm in length). Dracocephalum austriacum does not spread clonally and seeds lack adaptations for dispersal. It usually occurs in habitat patches where competition with other plants is low and competition with shrubs and trees may have a negative effect on performance (Olivier et al. 1995). Dracocephalum austriacum is listed as Vulnerable by the World Conservation Union (IUCN) and is protected under National, European and international legislation (Red List of French Endangered Flora, Habitats Directive, Bern Convention). Threats include pillaging, trampling and damage by grazing cattle (Bensettiti et al. 2002). Previous studies with this species have shown large population differences in adaptive genetic variation (Bonin et al. 2007) and positive correlations between population heterozygosity and stochastic population growth rate (Nicole 2005). The present field study was carried out in 7 of the 15 known French populations of D. austriacum. The study populations occur in the subalpine zone (1250 to 2000 m a.s.l.) of the Alps, on exposed, xeric, stony grasslands or heaths, preferentially on calcareous, thin soils (Lauber and Wagner 1998). Populations vary widely in size, density, habitat characteristics and soil composition.'

1.3 Data

We begin with reading in data. You'll have to set your own file path here. The data is in a .csv file included in the Supplementary Materials.

```
d <- read.csv(Appendix_A_Simple_data.csv)
```

You will notice that adults are stored at the beginning of the data set with values for size (measured in 2001) and sizeNext (measured in 2002). The number of seeds (fec.seed) and flowering status (fec.flower) were measured in 2001.

```
head(d)
```

```
  size sizeNext surv fec.seed fec.flower
1 3.09       NA    0       NA         NA
2 2.81       NA    0       NA         NA
3 4.46     6.07    1       15          1
4 1.68       NA    0       NA         NA
5 3.99       NA    0       NA         NA
6 4.07       NA    0       NA         NA
```

New recruits are stored at the end of the data frame and were only observed in the second survey (2002).

```
tail(d)
```

```
    size sizeNext surv fec.seed fec.flower
495   NA     1.78   NA       NA         NA
496   NA     1.01   NA       NA         NA
497   NA     1.68   NA       NA         NA
498   NA     1.44   NA       NA         NA
499   NA     1.08   NA       NA         NA
500   NA     1.62   NA       NA         NA
```

Note that we use NAs for missing or non-applicable data rather than 0 or some other indicator because this causes them to be automatically excluded from R's regression functions. For example, an individual that doesn't survive cannot grow or reproduce, hence values for sizeNext, fec.seed, and fec.flower are NA.

1.4 Analysis

1.4.1 Plots for data exploration

The model used here is described in detail in Example 1 in Section II of the main text. Using the notation there, this model can be written as:

$$n_{t+1}(z') = \int_\Omega K(z) n_t(z) dz = \int_\Omega [P(z',z) + F(z',z)] n_t(z) dz \tag{1.1}$$

$$P(z',z) = s(z)\ g(z',z) \tag{1.2}$$

$$F(z',z) = p_{flower}(z)\ f_{seeds}(z)\ p_{estab}\ f_{recruitsize}(z) \tag{1.3}$$

To understand the data, we plot survival, growth/shrinkage/stasis, number of seeds, and size of recruits as a function of our state variable, size.

```
par(mfrow=c(2,2),mar=c(4,4,2,1))
plot(d$size,jitter(d$surv),xlab="Size (t)", ylab="Survival to t+1") # jittered
plot(d$size,d$sizeNext,xlab="Size (t)",ylab="Size (t+1)")
plot(d$size,d$fec.seed,xlab="Size (t)",ylab="Seed Number")
hist(d$sizeNext[is.na(d$size)],main=,xlab=Recruit Size)
```

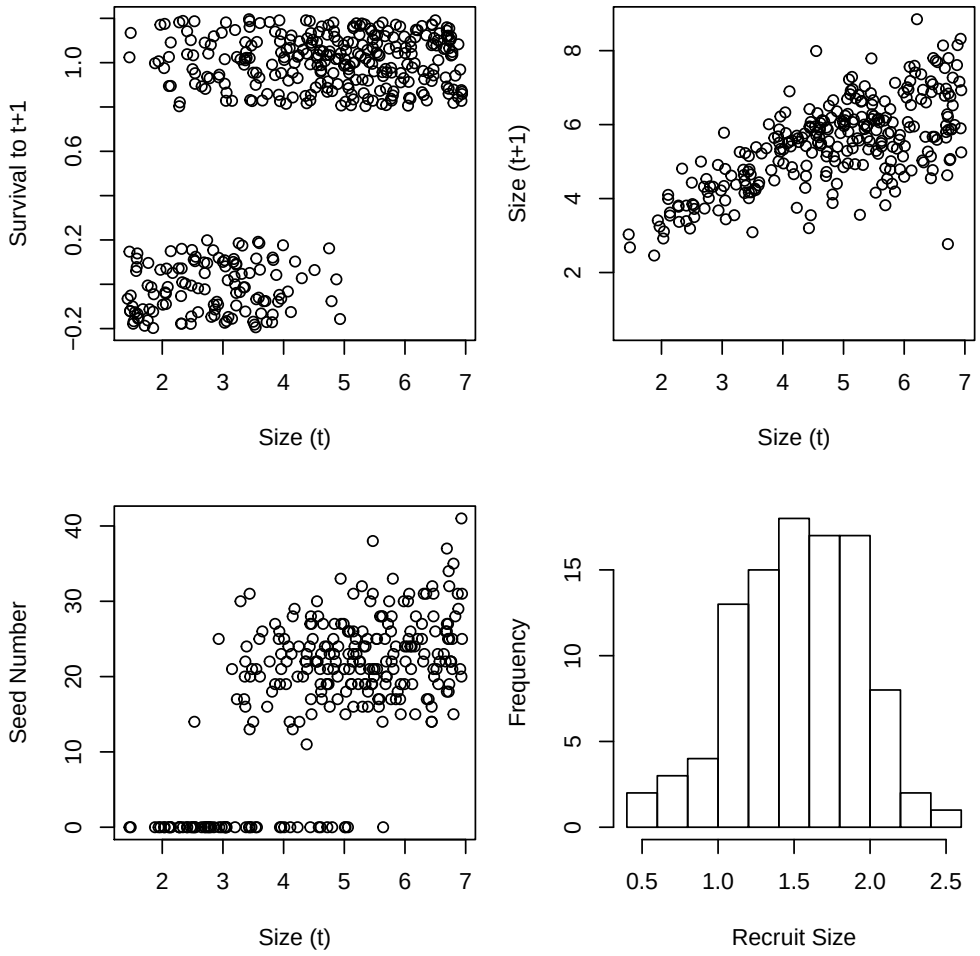

Figure 1.1: Exploration of vital rate data.

1.4.2 Build regressions for vital rate functions

We begin by setting up a data frame of the model parameters. These parameters will be estimated and recorded below.

```
params=data.frame(
   surv.int=NA,          # Intercept from logistic regression of survival
   surv.slope=NA,        # Slope from logistic regression of survival
```

```
    growth.int=NA,         # Intercept from linear regression of growth
    growth.slope=NA,       # Slope from linear regression of growth
    growth.sd=NA,          # Residual sd from the linear regression of growth
    seed.int=NA,           # Intercept from Poisson regression of seed number
    seed.slope=NA,         # Slope from Poisson regression of seed number
    recruit.size.mean=NA, # Mean recruit size
    recruit.size.sd=NA,   # Standard deviation of recruit size
    establishment.prob=NA # Probability of establishment
  )
```

Next, we build each of the vital rate regressions and store the coefficients.

```
 # 1. survival: logistic regression
 surv.reg=glm(surv~size,data=d,family=binomial())
 summary(surv.reg)
Call:
glm(formula = surv ~ size, family = binomial(), data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2275  -0.5815   0.2491   0.5597   2.0972

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.9646     0.4760  -8.329   <2e-16 ***
size          1.2897     0.1338   9.642   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 488.69  on 399  degrees of freedom
Residual deviance: 315.82  on 398  degrees of freedom
  (100 observations deleted due to missingness)
AIC: 319.82

Number of Fisher Scoring iterations: 5
 params$surv.int=coefficients(surv.reg)[1]
 params$surv.slope=coefficients(surv.reg)[2]
 # 2. growth: linear regression
 growth.reg=lm(sizeNext~size,data=d)
```

```
 summary(growth.reg)
Call:
lm(formula = sizeNext ~ size, data = d)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8037 -0.5434  0.0932  0.5741  2.6732

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.68135    0.19580   13.69   <2e-16 ***
size         0.57922    0.03902   14.84   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.8866 on 278 degrees of freedom
  (220 observations deleted due to missingness)
Multiple R-squared:  0.4421,        Adjusted R-squared:  0.4401
F-statistic: 220.3 on 1 and 278 DF,  p-value: < 2.2e-16
 params$growth.int=coefficients(growth.reg)[1]
 params$growth.slope=coefficients(growth.reg)[2]
 params$growth.sd=sd(resid(growth.reg))
```

For fecundity, note that we are pooling all individuals into this regression regardless of whether they flowered or not. A later exercise will be to explicitly model flowering probability, which necessary to deal with the many zeros that are observed in the plot for seed number in Fig. 1.1.

```
 # 3. seeds: Poisson regression
 seed.reg=glm(fec.seed~size,data=d,family=poisson())
 summary(seed.reg)
Call:
glm(formula = fec.seed ~ size, family = poisson(), data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.5338  -1.8830   0.0529   1.5375   4.9690

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.33234    0.06370   20.92   <2e-16 ***
size         0.30647    0.01164   26.34   <2e-16 ***
```

```
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2598.3  on 279  degrees of freedom
Residual deviance: 1840.1  on 278  degrees of freedom
  (220 observations deleted due to missingness)
AIC: 2942.2

Number of Fisher Scoring iterations: 5
```

```
params$seed.int=coefficients(seed.reg)[1]
params$seed.slope=coefficients(seed.reg)[2]
```

In the data frame, recruits are those individuals who have a value for sizeNext but not for size. We simply take the mean and standard deviation of their size for use in a normal distribution. We assume that offspring size is independent of maternal size so we only need to describe the distribution of offspring sizes in terms of its mean and variance.

```
# 4. size distribution of recruits
params$recruit.size.mean=mean(d$sizeNext[is.na(d$size)])
params$recruit.size.sd=sd(d$sizeNext[is.na(d$size)])
```

These data represent a single year's worth of data, hence establishment probability can be estimated by dividing the number of observed recruits by the number of seeds. the growth/survival measurements were taken in year t whereas the recruit sizes were measured in year t+1. Importantly, this calculation ignores all the processes that might lead to seed loss, however we make this assumption for the sake of simplicity in our illustration.

```
# 5. establishment probability
params$establishment.prob=sum(is.na(d$size))/sum(d$fec.seed,na.rm=TRUE)
```

Now, compare each of the vital rate models (red lines) to the data in Fig. 1.2.

Note that in practice, it is critical to check how well the vital rate models describe the data. Model selection is really the most important step in building an IPM because all inference derives from the quality of these regressions. Issues of model selection are discussed in the Section III.A of the main text and Appendix D.

```
par(mfrow=c(2,2))
xx=seq(0,8,by=.01) # sizes at which to evaluate predictions
plot(d$size,jitter(d$surv), xlab="Size (t)",ylab="Survival to t+1") # jittered
lines(xx,predict(surv.reg,
   data.frame(size=xx),type=response), col=red,lwd=3)
plot(d$size,d$sizeNext,xlab="Size (t)",ylab="Size (t+1)")
lines(xx,predict(growth.reg,data.frame(size=xx)),col=red,lwd=3)
plot(d$size,d$fec.seed,xlab="Size (t)",ylab="Seed Number (t)")
lines(xx,predict(seed.reg,
   data.frame(size=xx),type=response),col=red,lwd=3)
hist(d$sizeNext[is.na(d$size)],freq=FALSE,xlab="Recruit size",main=)
lines(xx,dnorm(xx,params$recruit.size.mean,
   params$recruit.size.sd), col=red,lwd=3)
```

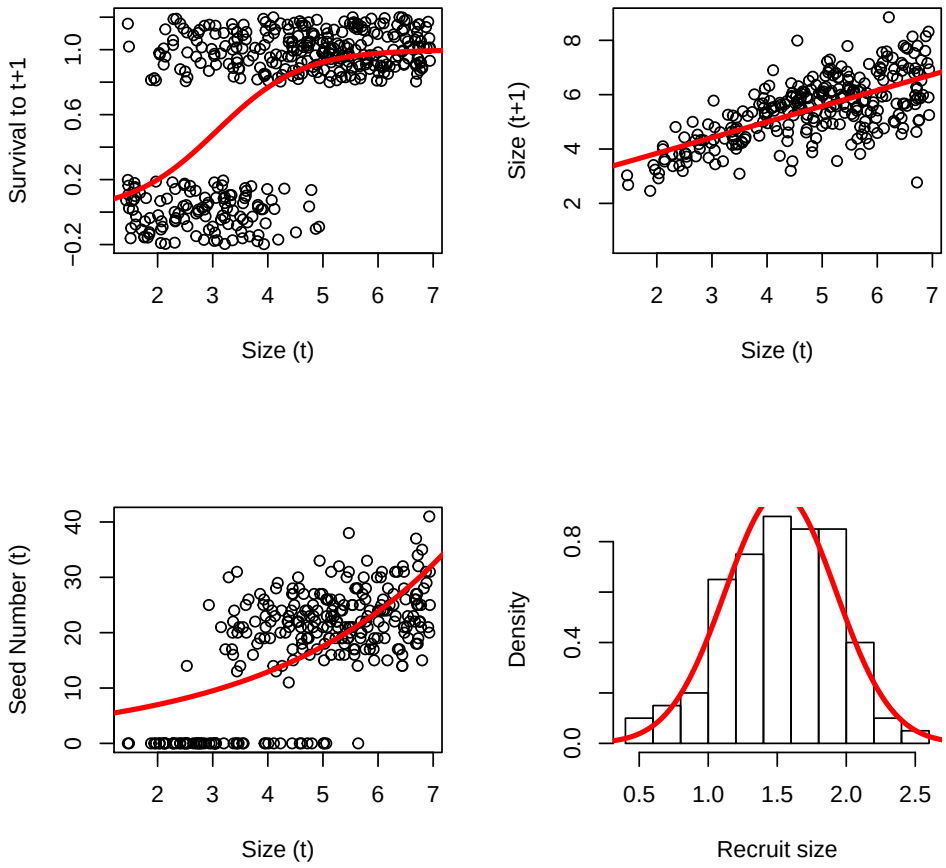

Figure 1.2: Compare vital rate regressions to data.

1.4.3 Define functions to describe life history

Each of the functions below represents one or more of the vital rates. These functions are used to build the IPM using output (the coefficients) from the regressions developed above.

These functions represent the modeler's decision about how to decompose the life cycle. In this very simple example, we model (1) survival, (2) growth, (3) seed number, (4) the seedling size distribution and (5) the establishment probability. In practice, we'd need to decide what regressions to build in part B in advance, but it's easier to digest this section after you've seen the regressions above, so fortunately we built all the right regressions already. In practice, sections 1.4.2 and 1.4.3 are iterative should inform one another. For example, the level of detail included in the life history models can be determined by the data availability and quality of the candidate vital rate regressions.

```
  # 1. survival probability function
s.x=function(x,params) {
   u=exp(params$surv.int+params$surv.slope*x)
   return(u/(1+u))
 }
# 2. growth function
g.yx=function(xp,x,params) {
   dnorm(xp,mean=params$growth.int+params$growth.slope*x,sd=params$growth.sd)
 }
# 3. reproduction function
f.yx=function(xp,x,params) {
   params$establishment.prob*
     dnorm(xp,mean=params$recruit.size.mean,sd=params$recruit.size.sd)*
     exp(params$seed.int+params$seed.slope*x)
 }
```

1.4.4 Make a kernel

In this section, we combine the vital rate functions to build the discretized IPM kernel, which we'll call the IPM matrix (e.g. shown in Fig. 2c in the main text).These steps are performed behind the scenes in the IPMpack package used in Appendices C-G for convenience, but we show them here for illustrative purposes. To integrate, we begin by defining the boundary points (b; the edges of the cells defining the matrix), mesh points (y; the centers of the cells defining the matrix and the points at which the matrix is evaluated for the midpoint rule of numerical integration), and step size (h; the widths of the cells). The integration limits (min.size and max.size) span the range of sizes observed in the data set, and then some.

```
min.size=.9*min(c(d$size,d$sizeNext),na.rm=T)
max.size=1.1*max(c(d$size,d$sizeNext),na.rm=T)
n=100 # number of cells in the matrix
```

```
b=min.size+c(0:n)*(max.size-min.size)/n # boundary points
y=0.5*(b[1:n]+b[2:(n+1)]) # mesh points
h=y[2]-y[1] # step size
```

Next, we make the IPM matrices. The function outer() evaluates the matrix at all pairwise combinations of the two vectors y and y and returns matrices representing the kernel components for growth and fecundity, respectively. For the numerical integration, we're using the midpoint rule (the simplest option) estimate the area under a curve. The midpoint rule assumes a rectangular approximation. The heights of the rectangles are given by the outer function and the width of the rectangles is h.

```
G=h*outer(y,y,g.yx,params=params) # growth matrix
S=s.x(y,params=params)            # survival
F=h*outer(y,y,f.yx,params=params) # reproduction matrix
P=G                               # placeholder; redefine P on the next line
for(i in 1:n) P[,i]=G[,i]*S[i]    # growth/survival matrix
K=P+F                             # full matrix
```

The result is 100 x 100 cell discretization of the kernel, K. For analyses, we can now apply the tools for matrix projection models (cf. Caswell 2001) to K.

1.4.5 Basic analyses

Analyses usually begin with obtaining the eigenvalues (λ) and eigenvectors (v-left; w-right) of the matrix. These are useful for understanding the asymptotic dynamics. The dominant eigenvalue gives the asymptotic population growth rate (lam). The right eigenvector gives the stable stage distribution and the left eigenvector gives the reproductive value, when normalized.

```
(lam <- Re(eigen(K)$values[1]))
[1] 1.013391
w.eigen <- Re(eigen(K)$vectors[,1])
stable.dist <- w.eigen/sum(w.eigen)
v.eigen <- Re(eigen(t(K))$vectors[,1])
repro.val <- v.eigen/v.eigen[1]
```

The eigen-things can be combined to obtain the sensitivity and elasticity matrices.

```
v.dot.w=sum(stable.dist*repro.val)*h
sens=outer(repro.val,stable.dist)/v.dot.w
elas=matrix(as.vector(sens)*as.vector(K)/lam,nrow=n)
```

Finally, we plot these results (stable size distribution, reproductive values, elasticity, sensitivity).

```
par(mfrow=c(2,3),mar=c(4,5,2,2))
image.plot(y,y,t(K), xlab="Size (t)",ylab="Size (t+1)",
   col=topo.colors(100), main="IPM matrix")
contour(y,y,t(K), add = TRUE, drawlabels = TRUE)
plot(y,stable.dist,xlab="Size",type="l",main="Stable size distribution")
plot(y,repro.val,xlab="Size",type="l",main="Reproductive values")
image.plot(y,y,t(elas),xlab="Size (t)",ylab="Size (t+1)",main="Elasticity")
image.plot(y,y,t(sens),xlab="Size (t)",ylab="Size (t+1)", main="Sensitivity")
```

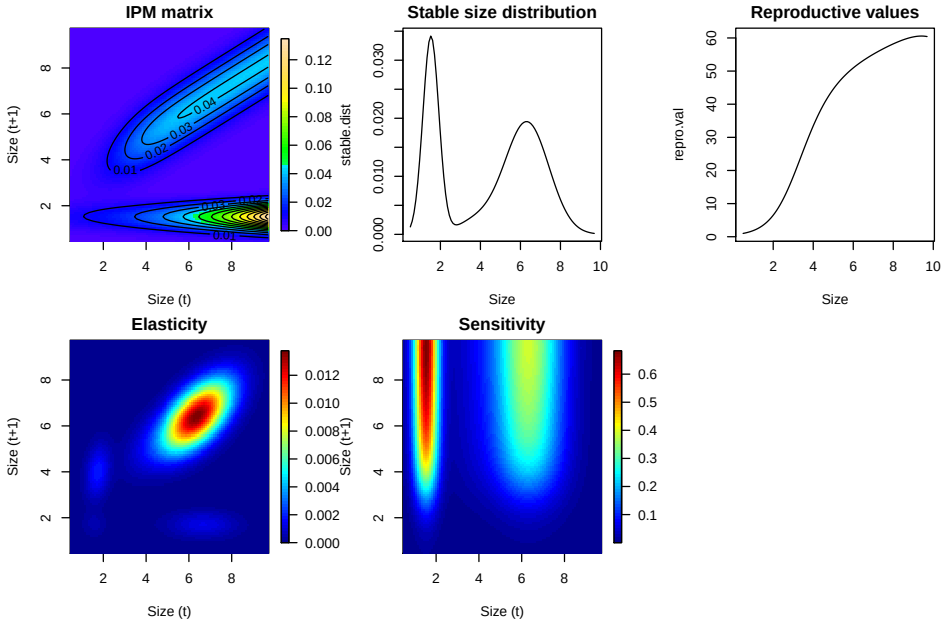

Figure 1.3: Basic model output

1.4.6 Improving the model

Typically, the workflow for building IPMs is iterative; based on preliminary analyses, we determine which important factors are missing or superfluous in the models. In this section, we make three improvements to the model built thus far: (1) we make the variance in growth among individuals a function of size (i.e. growth is more variable in larger individuals); (2) add a quadratic term for size to the growth function to capture the plateauing pattern in the growth data (Fig. 1.1b); (3) modify the fecundity function to include flowering probability, thereby recognizing that not all individuals reproduce.

Size Dependent Variance in Growth

First, we make the variance of the growth regression a function of size and obtain the value of λ. In the model above, we simply modeled the variance as constant, as seen in params$growth.sd. By looking at the growth data in Fig. 1.1b you might guess that the variance increases as a function of size. You can see this in the following plot, where we plot the absolute value of the residuals of

the growth regression against size.

```
plot(growth.reg$model$size,abs(resid(growth.reg)),
   xlab=size,ylab=residual)
```

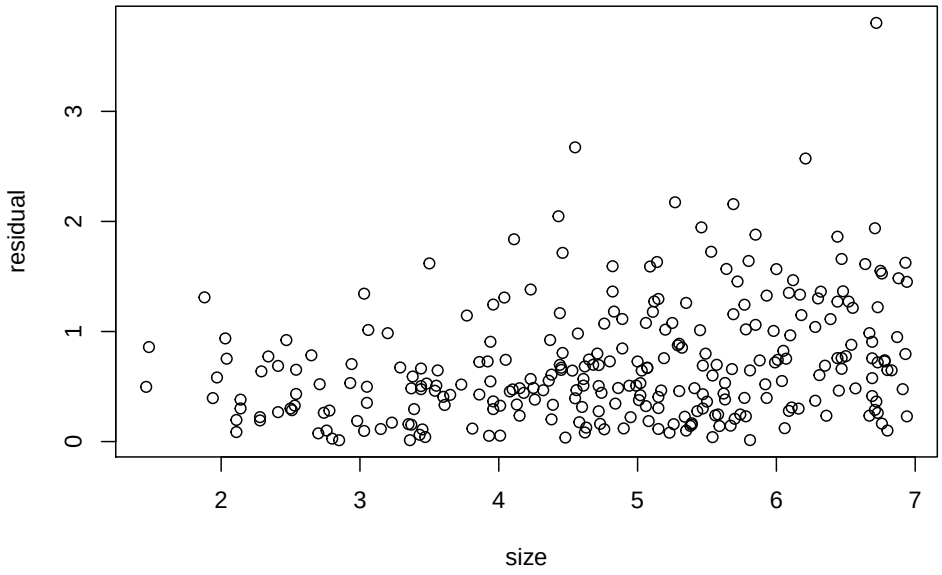

Figure 1.4: Increasing variance in growth

Incorporating nonuniform variance in the size distribution requires four steps:

(A) Build a regression on the residuals of the growth regression as a function of size. This is most easily done using generalized least squares (GLS). GLS allows us to simultaneously fit a model for the expected size at t+1 and the variance. For this example, we'll model the variance in growth as an exponential function: $\sigma^2(size) = \sigma^2 exp[2 * constant * size]$. See ?varExp for more details. The GLS model will estimate σ^2 and the constant, in addition to the intercept and slope used in the mean of the growth regression. gls() is in the nlme package, which you may need to install and load using install.packages('nlme') and library(nlme).

```
growth.reg=gls(sizeNext~size,weights=varExp(),na.action=na.omit, data=d)
summary(growth.reg)
```

```
Generalized least squares fit by REML
  Model: sizeNext ~ size
  Data: d
      AIC      BIC   logLik
  711.298 725.8085 -351.649
```

```
Variance function:
 Structure: Exponential of variance covariate
 Formula: ~fitted(.)
 Parameter estimates:
    expon
0.2935594

Coefficients:
                Value  Std.Error  t-value p-value
(Intercept) 2.3280286 0.14227273 16.36314       0
size        0.6585909 0.03301602 19.94762       0

 Correlation:
     (Intr)
size -0.945

Standardized residuals:
        Min          Q1         Med          Q3         Max
-3.29676786 -0.68003669  0.07382739  0.69180794  3.35550523

Residual standard error: 0.1664024
Degrees of freedom: 280 total; 278 residual
```

We plot this model, just to be sure it's reasonable. Note that the mean doesn't change much from the previous model; the effect of the GLS model is to modify the variance, which becomes apparent below (Fig. 1.6).

```
plot(d$size,d$sizeNext,main=Growth/Shrinkage/Stasis)
lines(xx,predict(growth.reg,data.frame(size=xx)),col=red,lwd=3)
```

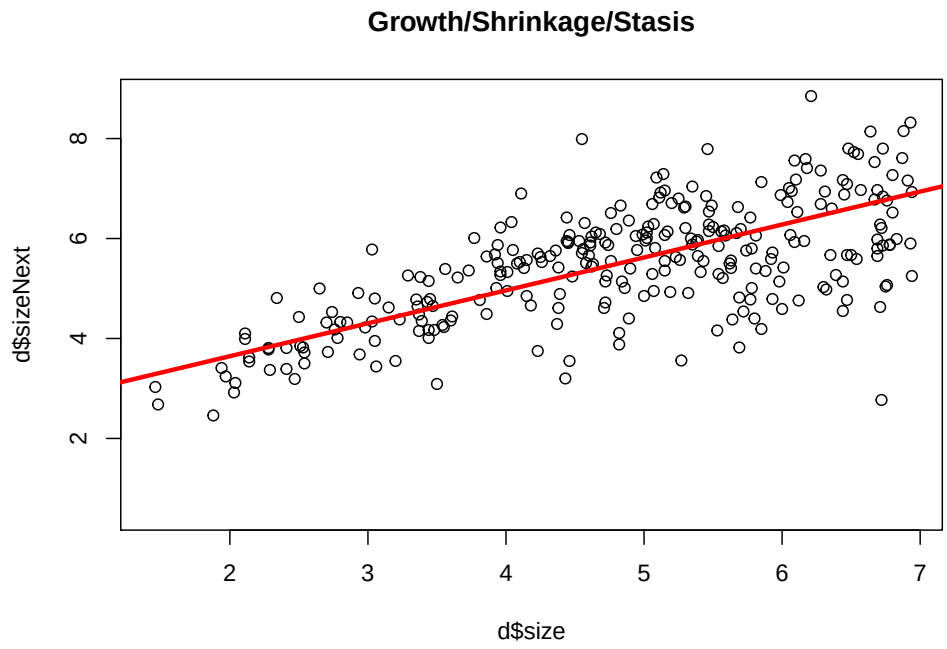

Figure 1.5: Size dependent variance in growth.

(B) Modify the params data frame (where we store all the coefficients used to build the IPM) to have coefficients called growth.sd.int and growth.sd.slope and set the values of those coefficients equal to the appropriate coefficients from part (A).

```
params$growth.int=coefficients(growth.reg)[1]
params$growth.slope=coefficients(growth.reg)[2]
params$growth.sigma2=summary(growth.reg)$sigma^2
params$growth.sigma2.exp=as.numeric(growth.reg$modelStruct$varStruct)
```

(C) Modify the growth function, g.xy(), to allow the standard deviation argument, sd, to be a function of size. This will follow a similar pattern to the argument for mean. It's probably easiest to redefine the function g.yx() so it works with code above to create IPMs.

```
g.yx=function(xp,x,params) {
   dnorm(xp,mean=params$growth.int+params$growth.slope*x,
     sd=sqrt(params$growth.sigma2*exp(2*params$growth.sigma2.exp*x)))
 }
```

(D) Rerun the code from sections 1.4.4 and 1.4.5 to obtain λ, sensitivities, elasticities, etc.

```
G=h*outer(y,y,g.yx,params=params) # growth matrix
S=s.x(y,params=params)            # survival
```

```
P=G                                  # placeholder; redefine P on the next line
for(i in 1:n) P[,i]=G[,i]*S[i]       # growth/survival matrix
F=h*outer(y,y,f.yx,params=params) # reproduction matrix
K=P+F                                # full matrix
(lam=Re(eigen(K)$values[1]))         # new population growth rate
```

```
[1] 0.980247
```

```
w.eigen=Re(eigen(K)$vectors[,1])
stable.dist=w.eigen/sum(w.eigen)
v.eigen=Re(eigen(t(K))$vectors[,1])
repro.val=v.eigen/v.eigen[1]
v.dot.w=sum(stable.dist*repro.val)*h
sens=outer(repro.val,stable.dist)/v.dot.w
elas=matrix(as.vector(sens)*as.vector(K)/lam,nrow=n)
```

We can plot these results and compare to Fig. 1.3.

```
par(mfrow=c(2,3),mar=c(4,5,2,2))
image.plot(y,y,t(K), xlab="Size (t)",ylab="Size (t+1)",
   col=topo.colors(100), main="IPM matrix")
contour(y,y,t(K), add = TRUE, drawlabels = TRUE)
plot(y,stable.dist,xlab="Size",type="l",main="Stable size distribution")
plot(y,repro.val,xlab="Size",type="l",main="Reproductive values")
image.plot(y,y,t(elas),xlab="Size (t)",ylab="Size (t+1)",main="Elasticity")
image.plot(y,y,t(sens),xlab="Size (t)",ylab="Size (t+1)", main="Sensitivity")
```

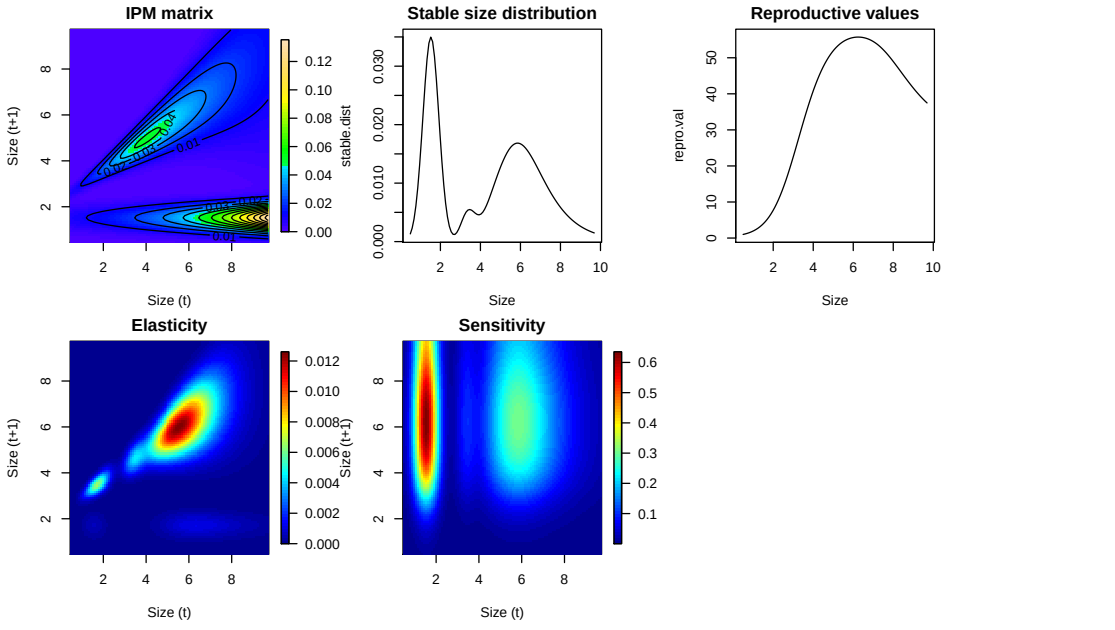

Figure 1.6: Basic model output with variance in growth function modeled as a linear function of size. Compare to Fig. 1.3.

Eviction

We begin by checking for unintentional eviction, wherein individuals transition beyond the size range chosen for the model, leading to inaccurate representation of the survival model (Williams et al. 2012). Because the growth function and offspring size distribution are Gaussian (Normal) probability densities, they range from -inf to +inf. Because of how we've defined the model, most of that probability for both functions falls between the minimum and maximum values of size that we've chosen for the model (0.45 and 9.735, respectively), but not all of it. If we ignore the parts of these densities that fall outside the integration limits, individuals are 'evicted from the model' and survival is incorrectly estimated (Williams et al. 2012).

To check for eviction, we plot the survival model and the column sums of the survival/growth (P) matrix. In Fig. 1.7, we see that the eviction occurs only for large individuals (size >9) because the column sums are lower than the survival models suggests that they should be.

```
plot(y,s.x(y,params),xlab="Size",type="l",
   ylab=Survival Probability,lwd=12)
points(y,apply(P,2,sum),col=red,lwd=3,cex=.1,pch=19)
```

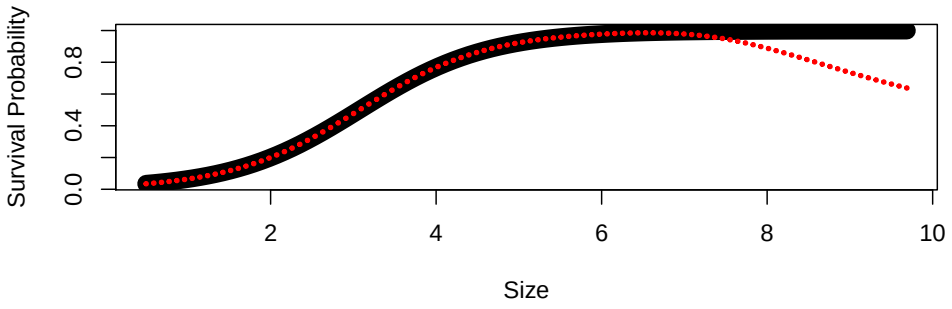

Figure 1.7: Comparison of predicted survival probability to observed values to test for unintentional eviction. The black line shows the fitted survival model. The red dot shows the column sums of the growth/survival matrix.

One way to correct for eviction is to return the evicted individuals to the cells at the boundaries where they were evicted. All individuals smaller than the lower integration limit are assigned to the smallest size class. This mainly affects the offspring size distribution; it sets a lower limit on offspring size. (Eviction at small sizes doesn't really affect the present model, but we mention it for completeness.) Similarly, all individuals larger than the upper integration limit can be assigned to the largest size class. This mainly affects the growth function, when individuals in the largest size class are still able to grow. A variety of approaches for correcting eviction are discussed and compared in (Williams et al. 2012).

To apply this approach, we need to modify the growth matrix before combining the growth/survival and fecundity matrices. In the following modification of the code to construct the matrix there

are two loops: one that corrects for eviction of offspring and one that corrects for eviction of large adults.

```
G=h*outer(y,y,g.yx,params=params) # growth matrix
S=s.x(y,params=params)
P=G
  # fix eviction of offspring
for(i in 1:(n/2)) {
   G[1,i]<-G[1,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
  # fix eviction of large adults
for(i in (n/2+1):n) {
   G[n,i]<-G[n,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
F=h*outer(y,y,f.yx,params=params) # reproduction matrix
K=P+F                             # full matrix
(lam=Re(eigen(K)$values[1]))      # new population growth rate
[1] 1.013069
```

We can re-plot the columns sums to check that this solution worked.

```
plot(y,s.x(y,params),xlab="Size",type="l",
   ylab=Survival Probability,lwd=12)
points(y,apply(P,2,sum),col=red,lwd=3,cex=.1,pch=19)
```

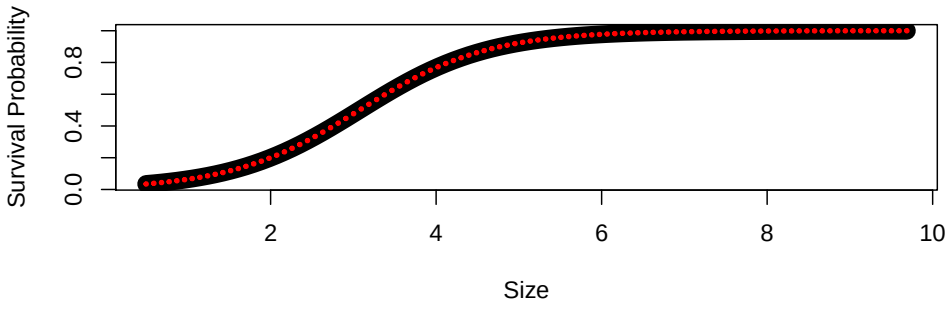

Figure 1.8: Comparison of predicted survival probability to observed values to test for unintentional eviction after using the ceiling approach to remove eviction. The black line shows the fitted survival model. The red dot shows the column sums of the growth/survival matrix.

Quadratic Growth Function

The next improvement to the IPM is to add a quadratic term to the growth regression to capture the slight downward curving pattern in growth (Fig. 1.2). To tell the growth regression to use a quadratic term, use I(size $\hat{2}$) as a predictor variable. For simplicity, we keep the size dependence of the growth variance from the previous section in the model, but note that we rerun the variance regressions since the residuals will change when the quadratic term is added to the model.

```
 growth.reg=gls(sizeNext~size+I(size^2),weights=varExp(), na.action=na.omit, data=d)
 summary(growth.reg)
Generalized least squares fit by REML
  Model: sizeNext ~ size + I(size^2)
  Data: d
       AIC      BIC    logLik
  698.7273 716.8474 -344.3636

Variance function:
 Structure: Exponential of variance covariate
 Formula: ~fitted(.)
 Parameter estimates:
    expon
0.3526462

Coefficients:
                 Value Std.Error   t-value p-value
(Intercept)  0.7802238 0.3208072  2.432065  0.0156
size         1.4981985 0.1704046  8.792009  0.0000
I(size^2)   -0.1007464 0.0204990 -4.914706  0.0000

 Correlation:
          (Intr) size
size      -0.972
I(size^2)  0.925 -0.985

Standardized residuals:
        Min          Q1         Med          Q3         Max
-3.26750298 -0.63630063  0.06390183  0.62856672  3.02974171

Residual standard error: 0.11715
Degrees of freedom: 280 total; 277 residual
```

```
params$growth.int=coefficients(growth.reg)[1]
params$growth.slope=coefficients(growth.reg)[2]
params$growth.sqrd=coefficients(growth.reg)[3]
params$growth.sigma2=summary(growth.reg)$sigma^2
params$growth.sigma2.exp=as.numeric(growth.reg$modelStruct$varStruct)
g.yx=function(xp,x,params) {
   dnorm(xp,
     mean=params$growth.int+params$growth.slope*x+params$growth.sqrd*x^2,
     sd=sqrt(params$growth.sigma2*exp(2*params$growth.sigma2.exp*x)))
 }
```

We rerun the code to build the matrix and obtain λ, sensitivities, elasticities, etc.

```
G=h*outer(y,y,g.yx,params=params) # growth matrix
S=s.x(y,params=params)            # survival
P=G                               # placeholder; redefine P on the next line
  # fix eviction of offspring
for(i in 1:(n/2)) {
   G[1,i]<-G[1,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
  # fix eviction of large adults
for(i in (n/2+1):n) {
   G[n,i]<-G[n,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
#for(i in 1:n) P[,i]=G[,i]*S[i]    # growth/survival matrix
F=h*outer(y,y,f.yx,params=params) # reproduction matrix
K=P+F                             # full matrix
(lam=Re(eigen(K)$values[1]))      # new population growth rate
[1] 0.9835344
w.eigen=Re(eigen(K)$vectors[,1])
stable.dist=w.eigen/sum(w.eigen)
v.eigen=Re(eigen(t(K))$vectors[,1])
repro.val=v.eigen/v.eigen[1]
v.dot.w=sum(stable.dist*repro.val)*h
sens=outer(repro.val,stable.dist)/v.dot.w
elas=matrix(as.vector(sens)*as.vector(K)/lam,nrow=n)
```

We can plot these results and compare to Figs. 1.3 and 1.6. Compared to the model in Fig. 1.6, the ridge in the matrix corresponding to growth in Fig. 1.9 now has a parabolic shape. The new

model prevents individuals from growing continuously and sets an (asymptotic) upper limit on size where the growth regression crosses the 1:1 line. Because individuals can not growth continuously in the model in Fig. 1.9, there is now more mass in the 'adult' portion of the stable stage distribution (sizes > 4), because there are no longer very large individuals in the population, who previously were producing a very large number of offspring in the model in Fig. 1.6. Not surprisingly, there is slightly greater sensitivity/elasticity to survival of individuals reaching sizes near 6, corresponding to the largest individuals in current model.

```
par(mfrow=c(2,3),mar=c(4,5,2,2))
image.plot(y,y,t(K), xlab="Size (t)",ylab="Size (t+1)",
   col=topo.colors(100), main="IPM matrix")
contour(y,y,t(K), add = TRUE, drawlabels = TRUE)
plot(y,stable.dist,xlab="Size",type="l",main="Stable size distribution")
plot(y,repro.val,xlab="Size",type="l",main="Reproductive values")
image.plot(y,y,t(elas),xlab="Size (t)",ylab="Size (t+1)",main="Elasticity")
image.plot(y,y,t(sens),xlab="Size (t)",ylab="Size (t+1)", main="Sensitivity")
```

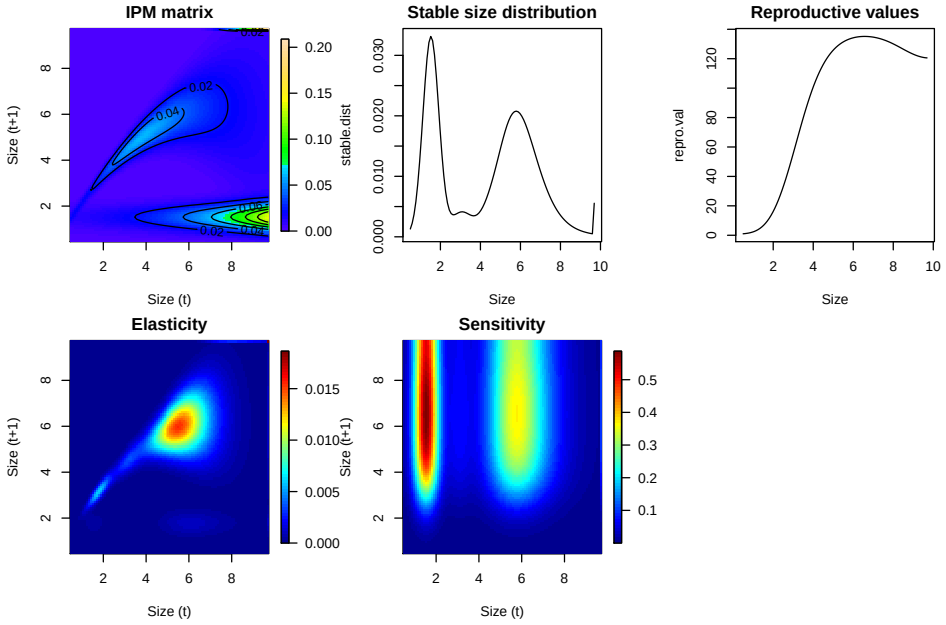

Figure 1.9: Basic model output with quadratic term in growth function. Compare to Figs. 1.3, 1.6.

Flowering Probability

The final improvement to the IPM is to incorporate a model for flowering probability. There are two issues in play here. One is that we know the biology and expect larger individuals to flower more. The other is that the data don't support the form of the original model. More generally, it is not a good idea to combine multiple processes into a single function, as this makes it hard to model the data. Consider the case where the plant does not flower each year, perhaps dependent on stress deriving from some environmental factor. We expect that larger plants are more likely to flower

because they can maintain a sufficient amount of stored resources to buffer against environmental variation. We might not know what triggers the decision to flower, but we can at least describe its size dependence by including the probability of flowering in the model. The flowering probability function will enter the reproduction kernel (called f.xy()), defined in section 6.4.2. The version of f.xy() used above simply assumes that all plants flower, so including the flowering probability function will reduce this number. (Alternatively, one could use a zero-inflated model, but since we have information on flowering, it is preferable to split up the distinct processes of flowering and reproductive output conditional on flowering.) Including a flowering probability model requires 5 steps:

(A) First, we write the flowering probability function. Flowering probability is most easily modeled using logistic regression, so this will be very similar to the survival function (s.xy) above. Call the function p.flower.x(). and store the slope and intercept of the flowering regression as params$flower.int and param$flower.slope, respectively, which are defined below.

```
p.flower.x=function(x,params) {
          u=exp(params$flower.int+params$flower.slope*x)
          return(u/(1+u))
 }
```

(B) Next, we modify the reproduction function (f.xy) to include the flowering probability function. This just amounts to multiplying the argument in f.xy by p.flower.x.

```
f.yx=function(xp,x,params) {
   p.flower.x(x,params)*
          params$establishment.prob*
          dnorm(xp,mean=params$recruit.size.mean,sd=params$recruit.size.sd)*
          exp(params$seed.int+params$seed.slope*x)
 }
```

(C) Now, we fit a logistic regression for flowering probability. See the data frame for binary flowering data under d$fec.flower. From this regression, we obtain a slope and intercept, and store these in the param vector.

```
flower.reg=glm(fec.flower~size,data=d,family=binomial())
summary(flower.reg)
```

```
Call:
glm(formula = fec.flower ~ size, family = binomial(), data = d)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.90763   0.05881   0.16048   0.39906   1.94232
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -6.5495     0.9564  -6.848 7.48e-12 ***
size          1.9081     0.2465   7.740 9.92e-15 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 285.68  on 279  degrees of freedom
Residual deviance: 144.30  on 278  degrees of freedom
  (220 observations deleted due to missingness)
AIC: 148.3

Number of Fisher Scoring iterations: 6
```

```
params$flower.int=coefficients(flower.reg)[1]
params$flower.slope=coefficients(flower.reg)[2]
```

(D) Update the regression for seed number to include only the individuals that flowered.

```
seed.reg=glm(fec.seed~size,data=d[d$fec.flower==1,],family=poisson())
summary(seed.reg)
```

```
Call:
glm(formula = fec.seed ~ size, family = poisson(), data = d[d$fec.flower ==
    1, ])

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.54532  -0.76716   0.03013   0.66809   2.98617

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.85617    0.07447  38.353  < 2e-16 ***
size         0.05076    0.01370   3.706  0.00021 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 250.71  on 221  degrees of freedom
```

```
Residual deviance: 236.90  on 220  degrees of freedom
  (220 observations deleted due to missingness)
AIC: 1339

Number of Fisher Scoring iterations: 4
```

```
params$seed.int=coefficients(seed.reg)[1]
params$seed.slope=coefficients(seed.reg)[2]
```

(E) Finally, we rerun the code from sections 1.4.4 and 1.4.5 to obtain λ, sensitivities, elasticities, etc., and plot the results.

```
G=h*outer(y,y,g.yx,params=params) # growth matrix
S=s.x(y,params=params)            # survival
P=G                               # placeholder; redefine P on the next line
  # fix eviction of offspring
for(i in 1:(n/2)) {
   G[1,i]<-G[1,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
  # fix eviction of large adults
for(i in (n/2+1):n) {
   G[n,i]<-G[n,i]+1-sum(G[,i])
   P[,i]<-G[,i]*S[i]
 }
# for(i in 1:n) P[,i]=G[,i]*S[i]    # growth/survival matrix
F=h*outer(y,y,f.yx,params=params) # reproduction matrix
K=P+F                             # full matrix
(lam=Re(eigen(K)$values[1]))      # new population growth rate
```

```
[1] 0.9765733
```

```
w.eigen=Re(eigen(K)$vectors[,1])
stable.dist=w.eigen/sum(w.eigen)
v.eigen=Re(eigen(t(K))$vectors[,1])
repro.val=v.eigen/v.eigen[1]
v.dot.w=sum(stable.dist*repro.val)*h
sens=outer(repro.val,stable.dist)/v.dot.w
elas=matrix(as.vector(sens)*as.vector(K)/lam,nrow=n)
```

```
par(mfrow=c(2,3),mar=c(4,5,2,2))
image.plot(y,y,t(K), xlab="Size (t)",ylab="Size (t+1)",
   col=topo.colors(100), main="IPM matrix")
contour(y,y,t(K), add = TRUE, drawlabels = TRUE)
plot(y,stable.dist,xlab="Size",type="l",main="Stable size distribution")
plot(y,repro.val,xlab="Size",type="l",main="Reproductive values")
image.plot(y,y,t(elas),xlab="Size (t)",ylab="Size (t+1)",main="Elasticity")
image.plot(y,y,t(sens),xlab="Size (t)",ylab="Size (t+1)", main="Sensitivity")
plot(y,predict(flower.reg,newdata=data.frame(size=y),type=response),
   xlab="Size (t)", ylab="Flowering probability",type=l)
```

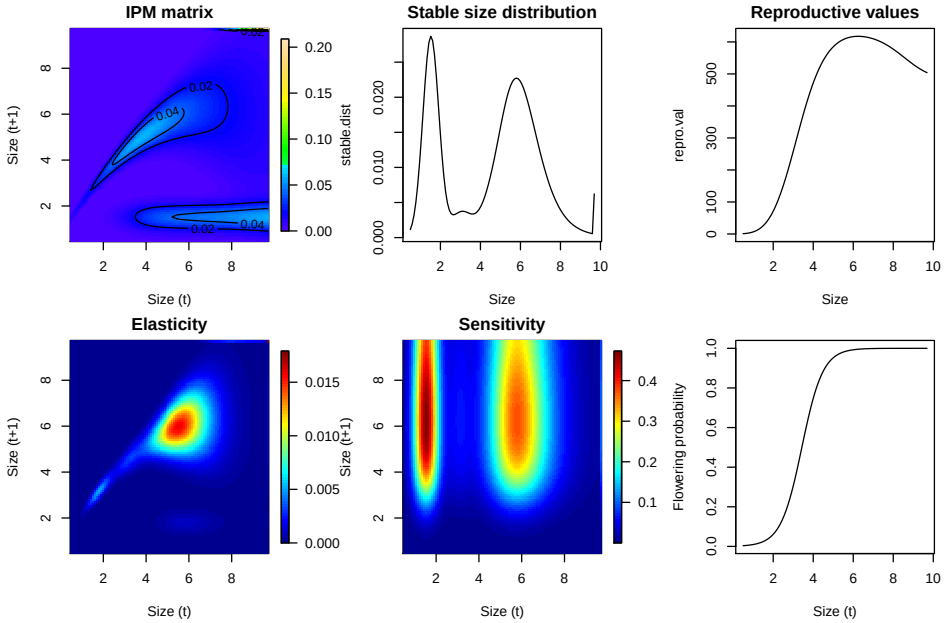

Figure 1.10: Basic model output with flowering probability function included. Compare to Figs. 1.3, 1.6, 1.9.

Note that λ decreases slightly compared to the model in Fig. 1.9, because fewer individuals reproduce. Compared to Fig. 1.9, the peak in the matrix and sensitivity is shifted toward individuals reaching a size near 6, where the flowering probability function asymptotes to 1 (Fig. 1.10). The stable size distribution shifts toward having more mass in established individuals (size>4) compared to Fig. 1.9. In the model in Fig. 1.9, small individuals reproduced 100% of the time; because fewer small individuals reproduce in the model in Fig. 1.10, a greater number of large individuals compensate for the lost reproductive output in the stable size distribution.

1.4.7 Diagnostics

In this section, we illustrate how to calculate some of the diagnostics for the kernel discussed in Section III of the main text. For these diagnostics, we rely on the R package 'IPMpack' (Metcalf et al. 2013), which conveniently implements the necessary calculations. While appendices C-G use

IPMpack for the full construction of IPMs, we illustrate here how the diagnostic tools can be used for an IPM matrix constructed outside the package.

Passage times

Next, we estimate the predicted time to reach critical a life history event. We begin with the life expectancy. This requires converting the matrix to an object of class 'IPMmatrix' that IPMpack can work with. Type *??IPMmatrix* for a description of the objects of class 'IPMmatrix'.

```
 library(IPMpack)
 Pmat = new("IPMmatrix", nDiscrete = 0, nEnvClass = 0,
          nBigMatrix = n, nrow = n, ncol = n, meshpoints = y,
          env.index = 0, names.discrete = )
 Pmat[, ] = P
 str(Pmat)
Formal class IPMmatrix [package "IPMpack"] with 7 slots
  ..@ .Data         : num [1:100, 1:100] 5.74e-14 5.44e-12 3.34e-10 1.31e-08 3.32e-07 ...
  ..@ nDiscrete     : num 0
  ..@ nEnvClass     : num 0
  ..@ nBigMatrix    : num 100
  ..@ meshpoints    : num [1:100] 0.496 0.589 0.682 0.775 0.868 ...
  ..@ env.index     : num 0
  ..@ names.discrete: chr ""
 (mle=meanLifeExpect(Pmat))
  [1]  1.067947  1.087713  1.113730  1.147586  1.191108
  [6]  1.246350  1.315576  1.401236  1.505916  1.632291
 [11]  1.783059  1.960869  2.168249  2.407523  2.680741
 [16]  2.989599  3.335372  3.718851  4.140281  4.599314
 [21]  5.094971  5.625612  6.188933  6.781973  7.401147
 [26]  8.042312  8.700842  9.371742 10.049768 10.729571
 [31] 11.405840 12.073441 12.727552 13.363778 13.978236
 [36] 14.567625 15.129262 15.661087 16.161653 16.630089
 [41] 17.066050 17.469655 17.841420 18.182189 18.493065
 [46] 18.775347 19.030470 19.259955 19.465367 19.648272
 [51] 19.810214 19.952689 20.077125 20.184874 20.277197
 [56] 20.355265 20.420153 20.472839 20.514208 20.545050
 [61] 20.566068 20.577879 20.581016 20.575939 20.563033
 [66] 20.542615 20.514941 20.480210 20.438568 20.390116
 [71] 20.334920 20.273015 20.204418 20.129137 20.047187
 [76] 19.958598 19.863433 19.761805 19.653884 19.539914
 [81] 19.420226 19.295242 19.165478 19.031550 18.894166
```

```
[86] 18.754116 18.612264 18.469530 18.326870 18.185258
[91] 18.045663 17.909030 17.776258 17.648185 17.525569
[96] 17.409079 17.299286 17.196655 17.101543 17.014202
```

Let's plot the result:

```
plot(y,meanLifeExpect(Pmat), xlab="Size (t)",ylab="Time")
```

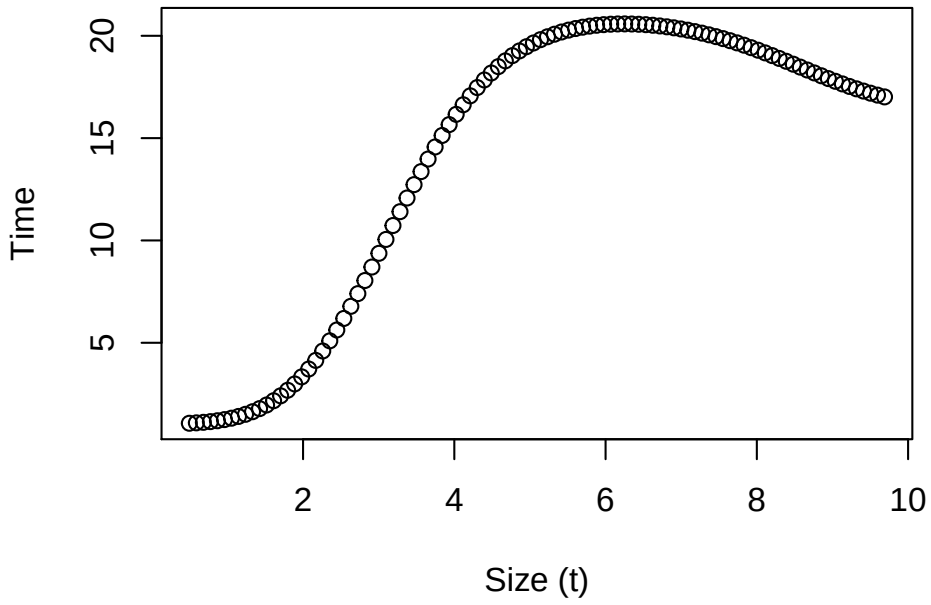

Figure 1.11: Life expectancy; the predicted mean lifespan of an individual of size z at time 0.

These life expectancy estimates seem reasonable for a long-lived perennial plant, so we continue to other diagnostics. One might also estimate the time to a variety of other life history events (cf. Caswell 2001, Chapter 5) to check that the model reasonably represents the species' biology.

Maximum size

The mean of the growth function crosses the 1:1 line in Fig. 1.12 around size = 6, this sets an *approximate* upper bound on the size to which individuals can grow. For example, if an individual is size 8 at time *t*, the expectation from this model is that it will be size 6.2 at time *t+1*. There's a small chance that it could grow to be larger than size 8, since the matrix has a nonzero values above the 1:1 line at size(t) = 8, but this is very unlikely. Since individuals larger than size 6 tend to shrink back toward size 6, we would expect that it should take a very long time to observe growth over a few consecutive years to reach the maximum size in the model (9.6). This illustrates why it

can be valuable to compare the growth model to the 1:1 line: where the mean of the growth model intersects 1:1 sets the approximate asymptotic size on individuals (Section III.A in the main text). This seems reasonable given the distribution of observed sizes shown in Fig. 1.12; most individuals are around size 6 while just a few grow to larger sizes. Note that if mortality is high, a species is slow growing, or the environment is variable, individuals may not reach this modeled maximum size.

```
par(mfrow=c(2,1),mar=c(4,5,2,2))
image.plot(y,y,t(P), xlab="Size (t)",ylab="Size (t+1)",
   col=topo.colors(100), main="IPM matrix")
contour(y,y,t(P), add = TRUE, drawlabels = TRUE)
abline(0,1,lwd=3,lty=2)
plot(density(d$sizeNext[!is.na(d$sizeNext)]),xlab=Size(t+1),
   main=Observed distribution of sizes)
```

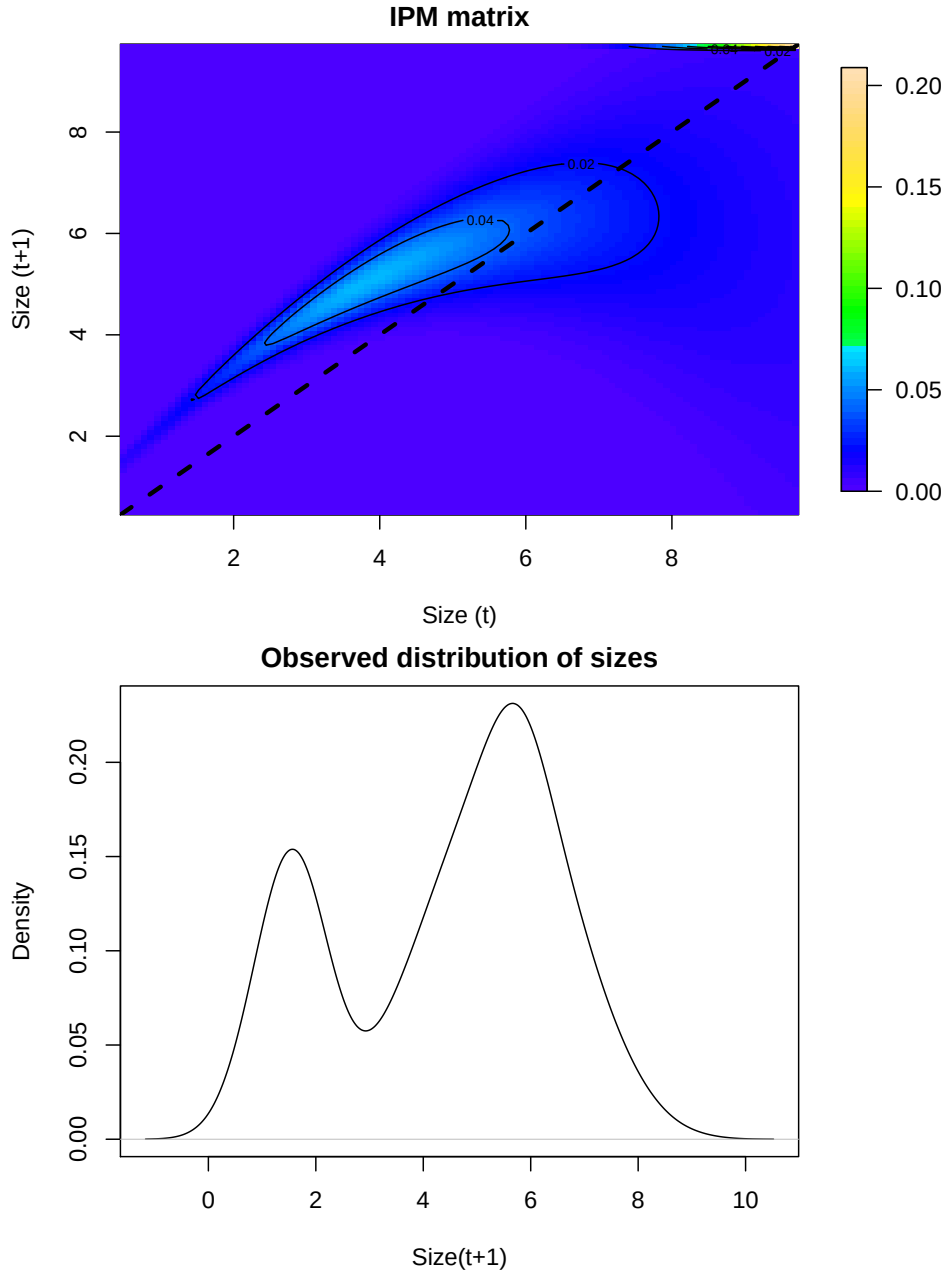

Figure 1.12: matrix with 1:1 line overlayed (dashed black line).

Transient Dynamics

Transient dynamics can be important if the population size structure is not near the stable size distribution. If interest lies in short term projections, considering transient dynamics may be particularly important for accurate predictions. Below we compare the asymptotic population growth rate to the inter-annual estimates to determine the time needed to reach a steady state. In this case, it requires only a few time steps, which means that the asymptotic predictions are sufficient for describing dynamics beyond a few years. The time scale for reaching the steady state can be estimated from the damping ratio (Caswell 2001, p. 95): the ratio of the dominant eigenvalue to the second largest one. Small values (near 1) mean slow transients; higher values means the dominant eigenvalue dictates the dynamics.

```
 (lam=Re(eigen(K)$values[1]))      # asymptotic growth rate
[1] 0.9765733
 (damp=Re(eigen(K)$values[1])/Re(eigen(K)$values[2])) # damping ratio
[1] 3.892877
 initial.pop=runif(100)  #random starting population structure
 initial.pop=initial.pop/sum(initial.pop)
 nyears=20
 size.dist=matrix(NA,n,nyears)
 lambda=rep(NA,nyears)
 xold=initial.pop
 for(i in 1:nyears){
          xnew=K%*%xold
    lambda[i]=sum(xnew)/sum(xold)
          size.dist[,i]=xnew
    xold=xnew
  }
 lambda
 [1] 1.0462480 0.9636261 0.9706708 0.9758530 0.9768217
 [6] 0.9767432 0.9766343 0.9765882 0.9765749 0.9765726
[11] 0.9765728 0.9765731 0.9765732 0.9765733 0.9765733
[16] 0.9765733 0.9765733 0.9765733 0.9765733 0.9765733
 (mean.lam=exp(mean(log(lambda))))
[1] 0.9789823
```

Note that the mean value of λ will be different than the asymptotic value during a period where transient dynamics are important.

1.5 Conclusions

Using this appendix as a guide, modelers should be able to construct basic stage structured IPMs from scratch. In appendices C-F, we illustrate how components of this process can automated using the R package IPMpack (Metcalf et al. 2013). For other examples of models similar to the one presented here, see Yule et al. (2013); Miller et al. 2009; Wallace et al. 2012 (crocodiles, though with similar vital rate models).

This appendix illustrates the common procedure of iteratively refining an IPM to capture greater complexity in vital rates. Many further refinements are possible. For example, one might wish to use more flexible forms of regression to accommodate nonlinear responses of vital rates (see Appendix G). Or, one might wish to include covariates, such as environmental predictors, that further refine predictive power of the vital rate regressions (see Appendix B). A more complex description of fecundity might also be desirable to disentangle the various processes that lead seed loss (see Appendix B; C).

1.6 Acknowledgements

Some of the code has been adapted from Nicole et al. (2011).

1.7 References

- Caswell, H. (2001). Matrix population models: construction, analysis, and interpretation. Sinauer, Sunderland, MA.
- Metcalf, C., McMahon, S.M., Salguero-Gomez, R. & Jongejans, E. (2013). IPMpack: an R package for integral projection models. Methods in Ecology and Evolution, 4, 195-200.
- Miller, T.E.X., Louda, S.M., Rose, K.A. & Eckberg, J.O. (2009). Impacts of insect herbivory on cactus population dynamics: experimental demography across an environmental gradient. Ecological Monographs, 79, 155-172.
- Nicole, F., Dahlgren, J.P., Vivat, A., Till-Bottraud, I. & Ehrlen, J. (2011). Interdependent effects of habitat quality and climate on population growth of an endangered plant. Journal of Ecology, 99, 1211-1218.
- Wallace, K., Leslie, A. & Coulson, T. (2012). Re-evaluating the effect of harvesting regimes on Nile crocodiles using an integral projection model (M. Boots, Ed.). Journal of Animal Ecology, 82, 155-165.
- Williams, J.L., Miller, T.E.X & Ellner, S.P. (2012). Avoiding unintentional eviction from integral projection models. Ecology, 93, 2008-2014.

- Yule, K.M., Miller, T.E.X. & Rudgers, J.A. (2013). Costs, benefits, and loss of vertically transmitted symbionts affect host population dynamics. Oikos, in press.

Chapter 2

Appendix B: Examples of fecundity models

Contact: Eelke Jongejans (e.jongejans@science.ru.nl)

Abstract

In this appendix, we delve into models of fecundity; reproduction is often the most complex and difficult part of a life cycle to capture. We begin by building an IPM kernel for the perennial plant *Succisa pratensis* using the R package *IPMpack*. The IPM consists of a survival-growth kernel, a clonal propagation kernel and a sexual reproduction kernel. We model reproduction of this plant with variable complexity and detail, and evaluate whether the choice of fecundity model affects model output. The analyses show only small differences in projected population growth rates, but more complex models (i.e. with more detailed steps in the reproduction pathway) give more insight into which vital rates matter for population dynamics. We also compare alternative state variables, and see that the constructed IPMs can differ considerably, also in population growth rate and elasticity patterns.

Figure 2.1: Devil's Bit Scabiosa, *Succisa pratensis*. Picture by Eelke Jongejans

2.1 Introduction

The goal of this appendix is to lead the user through the process of modeling fecundity in Integral Projection Models using the R package *IPMpack* (Metcalf et al. 2013). First we will show how IPMs are built with *IPMpack*, followed by two exercises in which the complexity of the reproduction kernel and the state variable are varied. After completing these exercises, users will be able to construct various fecundity objects in *IPMpack*, and create an IPM matrix from fecundity, clonality, survival, and growth objects.

2.2 Study system/objectives

For this appendix we use data on the population dynamics of *Succisa pratensis* a plant of nutrient-poor, species-rich meadows in the Netherlands (Jongejans & de Kroon 2005). Specifically, *Succisa pratensis* Moench (Dipsacaceae, devil's bit scabiosa; Figure 2.1) is a perennial, polycarpic, rosette-forming plant characteristic of *Cirsio dissecti-Molinietum* communities (Adams 1955; Schaminee et al. 1996, Hooftman & Diemer 2002). Stem leaves grow in alternating pairs and persist for almost a year. Flowers are produced on flowering stems from late July until October. Censuses were conducted in late summer, thus the IPM will be structured on the basis of a pre-reproductive census. We use a subset of the demographic data collected at the 'Bennekomse Meent' site in the Netherlands. In our IPMs, the rosette is the individual unit. The formation of side-rosettes is therefore modeled as the clonal production of new 'individuals'.

2.3 Data

We begin by reading in data. The data file we use is available as part of the R package *IPMpack* (version 2.0 onward):

```
require(IPMpack)
data(dataIPMpackSuccisa)
Sp1 <- dataIPMpackSuccisa
head(Sp1)
```

```
      size sizeNext      stage  stageNext surv
1 6.282965 4.801939 continuous continuous    1
2 3.472842 5.376433 continuous continuous    1
3 4.902599 4.624504 continuous continuous    1
4 6.074348 6.588023 continuous continuous    1
5 7.218641 6.588023 continuous continuous    1
6 6.937394 6.703946 continuous continuous    1
  offspringNext fec1Bolt fec2Stem fec3Head fec1BoltNext
1          <NA>        0       NA       NA            1
2          <NA>        0       NA       NA            0
3          <NA>        0       NA       NA            0
4          <NA>        0       NA       NA            1
5          <NA>        0       NA       NA            1
6          <NA>        1        1        4            1
  fec2StemNext fec3HeadNext cloning clonesNext
1            2          1.5       0         NA
2           NA           NA       0         NA
3           NA           NA       0         NA
4            2          3.0       0         NA
5            2          3.5       0         NA
6            1          4.0       0         NA
```

In this data set, 'size' is defined as the log of the product of the number of leaves times the length of the longest leaf in a rosette. The variable 'sizeNext' is the size of the same individual one year later. 'surv' indicates whether the individual survived the year (1) or not (0). The variable 'fec1Bolt' indicates whether an individual produces any flowering stems at the beginning of the year. Conditional on the presence of at least one stem, 'fec2Stem' gives the number of flowering stems, and 'fec3Head' gives the number of inflorescences (otherwise, these variables contain 'NA'). The variable 'cloning' indicates whether an individual cloned (1) or not (0), while 'clonesNext' indicates the number of clones (which turns out to only have a value of 1).

To understand the data, we plot survival, growth/shrinkage/stasis, number of seeds, and size of recruits in Fig. 2.2

```
par(mfrow=c(3,2),mar=c(4,4,2,1))
plot(Sp1$size,jitter(Sp1$surv),xlab="Size (t)", ylab="Survival to t+1") # jittered
plot(Sp1$size,Sp1$sizeNext,xlab="Size (t)",ylab="Size (t+1)")
plot(Sp1$size,jitter(Sp1$fec1Bolt),xlab="Size (t)", ylab="Flowering probability") # jitte
plot(Sp1$size,Sp1$fec2Stem,xlab="Size (t)",ylab="Stem Number")
plot(Sp1$size,Sp1$fec3Head,xlab="Size (t)",ylab="Flower Head Number")
hist(Sp1$sizeNext[is.na(Sp1$size)],main=,xlab=Recruit Size)
```

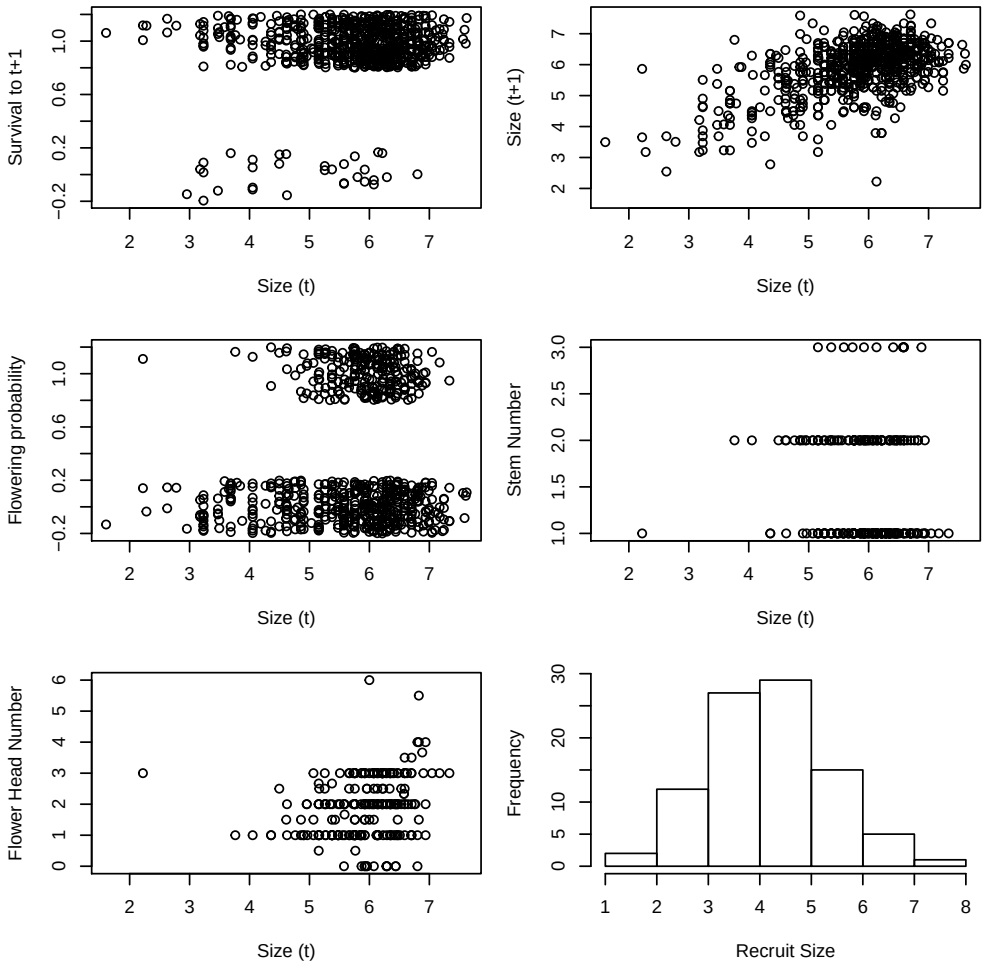

Figure 2.2: Exploration of vital rate data.

2.4 Analysis

We begin by describing equations that will govern the IPMs. The first equation below describes how the stage frequency of individuals in time t changes to a stage frequency of individuals in time t as a function of the kernel $\boldsymbol{K}$. This kernel contains three demographic processes: changes in size/stage conditional on survival, described by the sub-kernel $\boldsymbol{P}$, the per-capita contribution of reproductive individuals to new recruits from seed, described by the sub-kernel $\boldsymbol{F}$, and the per-

capita contribution of reproductive individuals to new recruits from clonal reproduction, described by the sub-kernel $\boldsymbol{C}$.

$$n_{t+1}(z') = \int_{\Omega} K(z) n_t(z) dz = \int_{\Omega} [P(z',z) + F(z',z) + C(z',z)] n_t(z) dz \tag{2.1}$$

The sub-kernel $\boldsymbol{P}$ is further decomposed into two vital rates: the probability that individuals of within a given stage (*i.e.* continuous or discrete) survive as a function of their size z in time (t), and the probability that, having survived in that time interval, an individual of a size z in time t will grow, stay the same size, or shrink to size z' in time $t+1$.

$$P(z',z) = surv(z)\ growth(z',z) \tag{2.2}$$

Likewise, the sub-kernel $\boldsymbol{F}$ is a function of a series of conditional demographic processes, or vital rates. As it is typically the case in the $\boldsymbol{F}$ sub-kernel, more vital rates will be modeled than in the sub-kernel $\boldsymbol{P}$ due to the slightly more complicated nature of reproduction in comparison to survival/changes in size of already established individuals. The processes to consider here are the probability of reproduction *fec1Bolt*; conditional on being reproductive, the number of stems produced, *fec2Stem*; the number of flower heads per stem, *fec3Head*; the number of seeds per flower head, *seedsPerHead*; the probability that a germinant establishes, *seedlingEstablishmentRate*; and the size distribution of established recruits, *recruitSize*.

$$F(z',z) = fec1Bolt(z) fec2Stem(z)\ fec3Head(z)\ seedsPerHead \\ seedlingEstablishmentRate\ recruitSize(z')$$

The sub-kernel $\boldsymbol{C}$ is also a function of a series of conditional demographic processes. Clonal reproduction is product of the probability of clonal reproduction *clonal*; conditional on being clonally reproductive, the number of clones produced, *clonesNext*; and the size distribution of clones, *cloneSize*.

$$C(z',z) = clone(z)\ clonesNext(z)\ cloneSize(z')$$

Before making survival, growth, and fecundity objects (for the construction and analysis of IPMs), we first define a size axis and the size range. The mesh points used for integration of the *Succisa pratensis* IPM will be based on this size axis.

```
x<-seq(from=0,to=10,length=1001)
x0<-data.frame(size=x,size2=x*x)
minSize<-min(Sp1$size,na.rm=T)
maxSize<-max(Sp1$size,na.rm=T)
```

2.4.1 Survival and growth kernel

We rapidly move through the process of building a survival and growth kernel, since this process is detailed further in the vignette for IPMpack (type *vignette("IPMpack_Vignette")*), and Appendices C,D,E,G. IPMpack is written in object-oriented code, using S4 objects. IPMpack contains defined classes for growth, survival and fertility objects, and associated methods that allow the user to build IPM objects. Thus the steps for building an IPM include building (1) a survival object; (2) a growth object; (3) a fecundity object. (4) These objects are combined to make an IPM object, which (5) can then be analyzed to produce various summaries of population dynamics.

The first step in constructing an IPM with *IPMpack* is a survival analysis. We use the function 'survModelComp' to explore whether survival is related to size, as illustrated in figure 2.3.

```
survModelComp(dataf = Sp1, makePlot = TRUE, legendPos = "bottomright",
   mainTitle = "Survival")
```

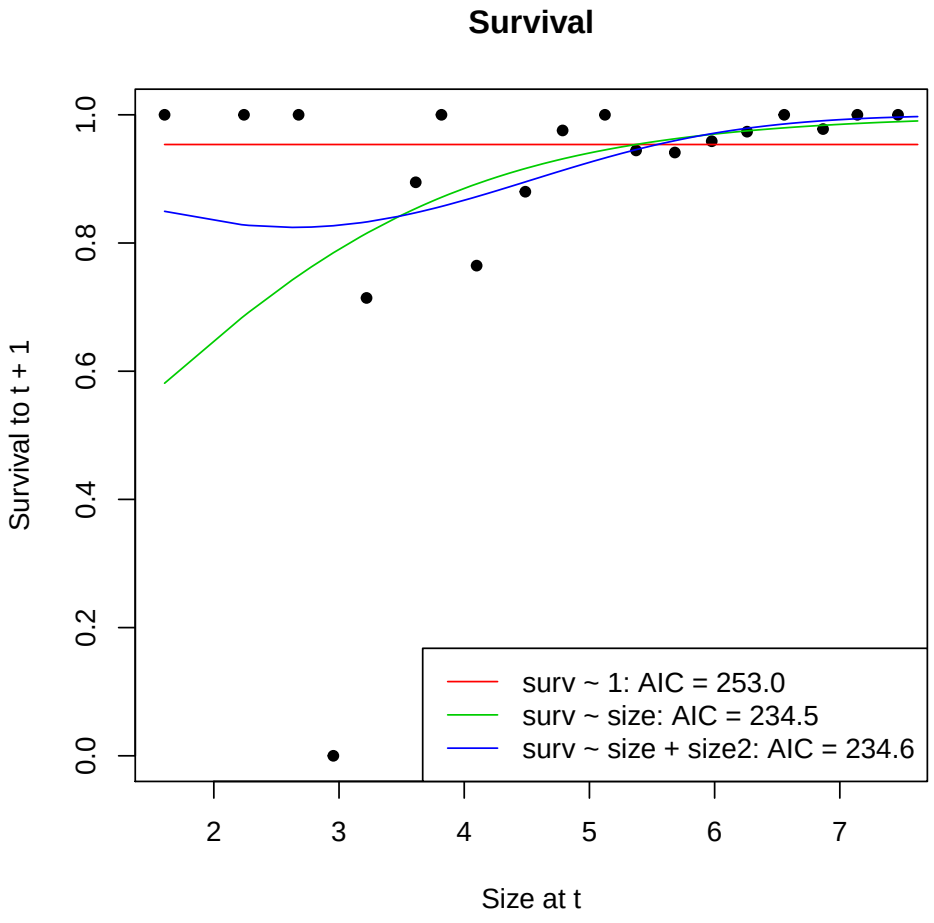

Figure 2.3: Annual survival of *Succisa pratensis* rosettes.

Based on this simple analysis we select the following survival model since it has the lowest AIC value:

```
so<-makeSurvObj(Sp1,surv~size)
```

We next model growth, conditional on survival. Here, 'growth' is the process relating size in year t+1 to size in year t. We use the following to make figure 2.4:

```
growthModelComp(dataf = Sp1, makePlot = TRUE, legendPos = "bottomright",
   mainTitle = "Growth")
```

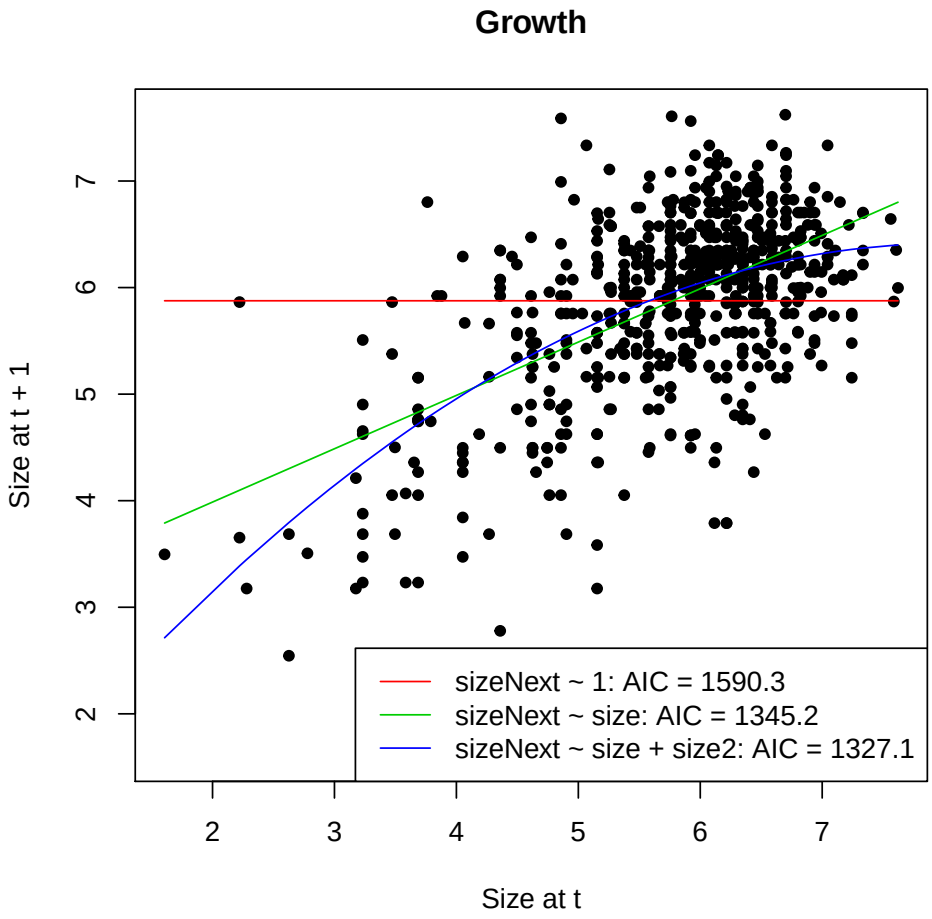

Figure 2.4: Annual 'growth' of *Succisa pratensis* rosettes.

Based on this simple model comparison, we select the following growth model:

```
go<-makeGrowthObj(Sp1,sizeNext~size+size2)
```

With these survival and growth objects in hand, we build a survival/growth (P) matrix.

```
Pmatrix<-makeIPMPmatrix(survObj=so,
```

```
                    growObj=go,
                    minSize=minSize,
                    maxSize=maxSize)
```

We plot this P-matrix using the 'image.plot' function of the *fields* package:

```
require(fields)
image.plot(Pmatrix@meshpoints,
           Pmatrix@meshpoints,
           t(Pmatrix),
           main = "Pmatrix: survival and growth",
           xlab = "Size at t",
           ylab = "Size at t+1")
abline(0,1,lty=2,lwd=3)
```

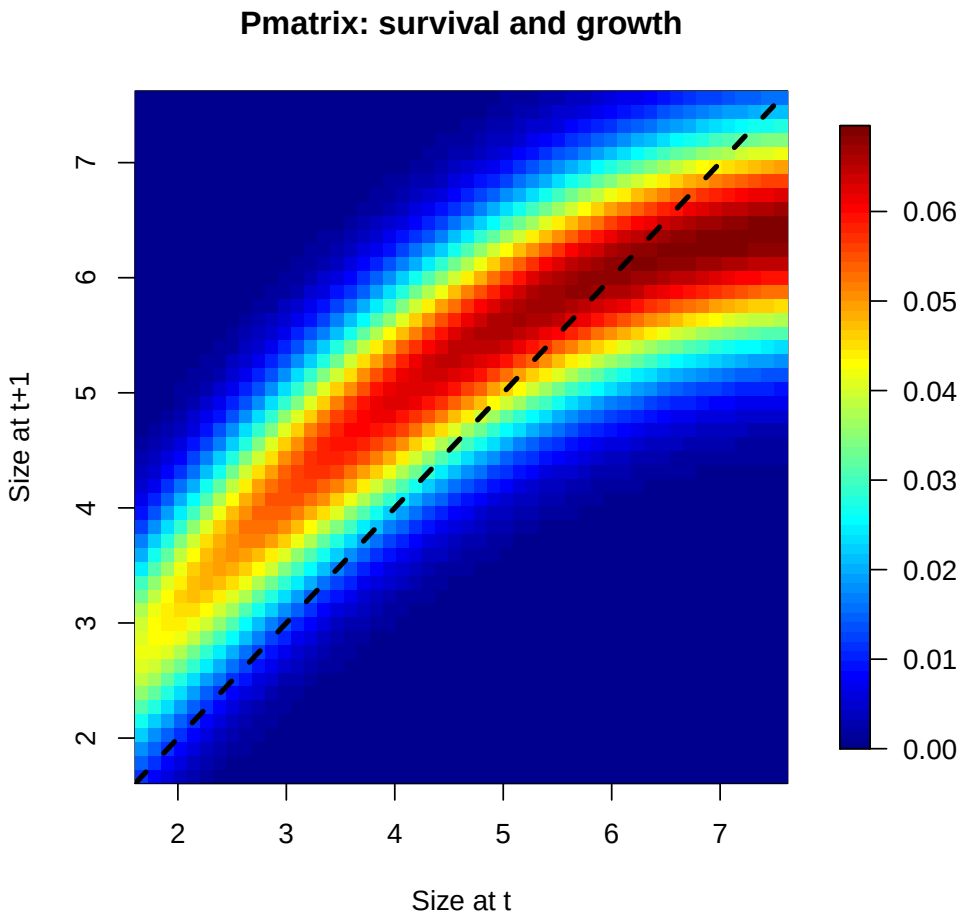

Figure 2.5: Survival and growth kernel.

The dashed 1:1 line in Fig.2.5 indicates stasis; individuals that fall along this line neither grow

nor shrink. Individuals above the dashed line grow, hence we can see that the location where the growth model crosses 1:1 sets an approximate maximum size for individuals. In this example, the approximate maximum size is 6. Although individuals can grow to larger sizes (some probability mass exists above the 1:1 line for large individuals), they will tend to shrink back to smaller sizes at subsequent times.

Running *IPMpack* diagnostics shows whether survival, life expectancy, and populations structure change with an increase in the number of bins or an increase in the size range (Fig. 2.6). The output illustrates that our P matrix summarizes the survival and growth part of the life cycle adequately (*i.e.*, in sufficient resolution), because the predictions do not change based on our choice of discretization.

```
diagnosticsPmatrix(Pmatrix, growObj=go, survObj=so, correction="constant")
[1] "Range of Pmatrix is "
[1] 8.999632e-13 6.958165e-02
[1] "Please hit any key for the next plot"
```

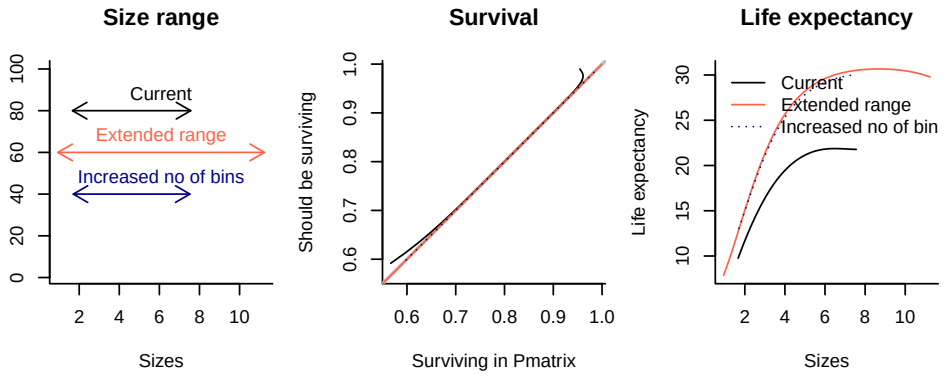

Figure 2.6: P-matrix diagnostics.

However, eviction does occur past the maximum size we selected. This can be best seen in the middle panel of Fig. 2.6, where the survival predicted from the column sums of the P matrix do not match the survival probabilities from the survival model. Thus we increase the size range and rebuild the P matrix. The command correction='constant' prevents 'eviction' from occurring at the extremes of the size range (see Section III.B in the main text).

```
minSize<-min(Sp1$size,na.rm=T)-1
maxSize<-max(Sp1$size,na.rm=T)+2
Pmatrix<-makeIPMPmatrix(survObj=so, growObj=go, minSize=minSize,
   maxSize=maxSize,correction="constant")
```

2.4.2 Fecundity kernel

The fecundity component of an IPM requires analysis of each step of the process of reproduction. Here we start with the first step: whether or not individuals flower in year t (a binomial response). We reemphasize that the population was censused during flowering, and thus we construct the population model based on a pre-reproductive census. Since *IPMpack* does not have a fecundity model comparison function yet, we must perform model comparison manually:

```
fo1<-makeFecObj(Sp1, Formula=fec1Bolt~1, Family = "binomial") # Intercept only model
fo2<-makeFecObj(Sp1, Formula=fec1Bolt~size, Family = "binomial")
fo3<-makeFecObj(Sp1, Formula=fec1Bolt~size+size2, Family = "binomial")
```

We plot these models for comparison in Fig. 2.7.

```
fs <- order(Sp1$size)
fs.fec <- (Sp1$fec1Bolt)[fs]
fs.size <- (Sp1$size)[fs]
pfz <- tapply(fs.size, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
ps <- tapply(fs.fec, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
plot(as.numeric(pfz), as.numeric(ps), pch = 19, cex=2, col="blue",ylim=c(0,1),
   xlab="size", ylab="proportion flowering", main="")
y0<-predict(fo1@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="red")
y0<-predict(fo2@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="green")
y0<-predict(fo3@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="blue")
legend("topleft", legend = sprintf("%s: %s = %.1f",c("1","size","size+size2"),
   c("AIC"),c(AIC(fo1@fitFec[[1]]),AIC(fo2@fitFec[[1]]),AIC(fo3@fitFec[[1]]))),
   col = c(2:4),lty = 1, xjust = 1, bg = "white")
```

The 'fec1Bolt $\sim$ size+size2' regression model best fits the data. It is a bit difficult to understand why there would be an optimal size for flowering (at size 6), a point to which we return later.

The next step is an analysis of the number of stems, conditional on the fact that an individual has produced at least one flowering stem. Thus, zeros are absent in the variable fec2Stem. We transform the 'fec2Stem' data by subtracting 1, so that the transformed data can be modeled as a Poisson distribution. This is achieved by setting 'Transform="-1"'; IPMpack will take care of transformation and back transformation from here. Note that 'makeFecObj()' makes all the components of the fecundity model simultaneously, so we simply add the specifications for the stem number model of the arguments used in the last step for flowering probability. That is, the

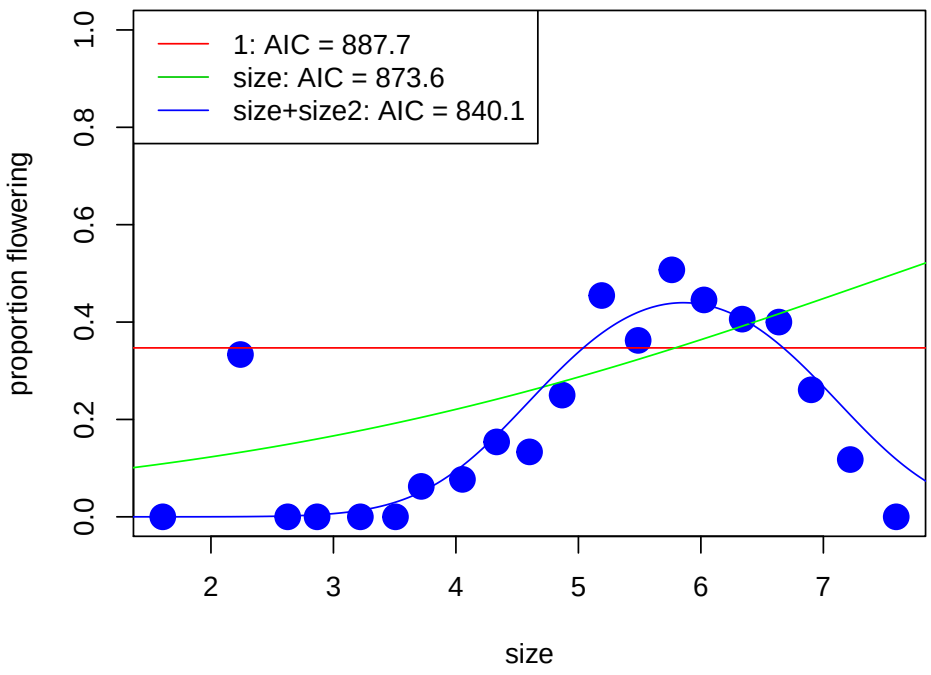

Figure 2.7: Logistic regression models of the probability of flowering.

arguments 'Formula', 'Family', and 'Transform', now have two arguments each, where the second argument corresponds to the stem number model.

```
fo1<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~1),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
fo2<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~size),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
fo3<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~size+size2),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
```

We plot these models of the number of stems:

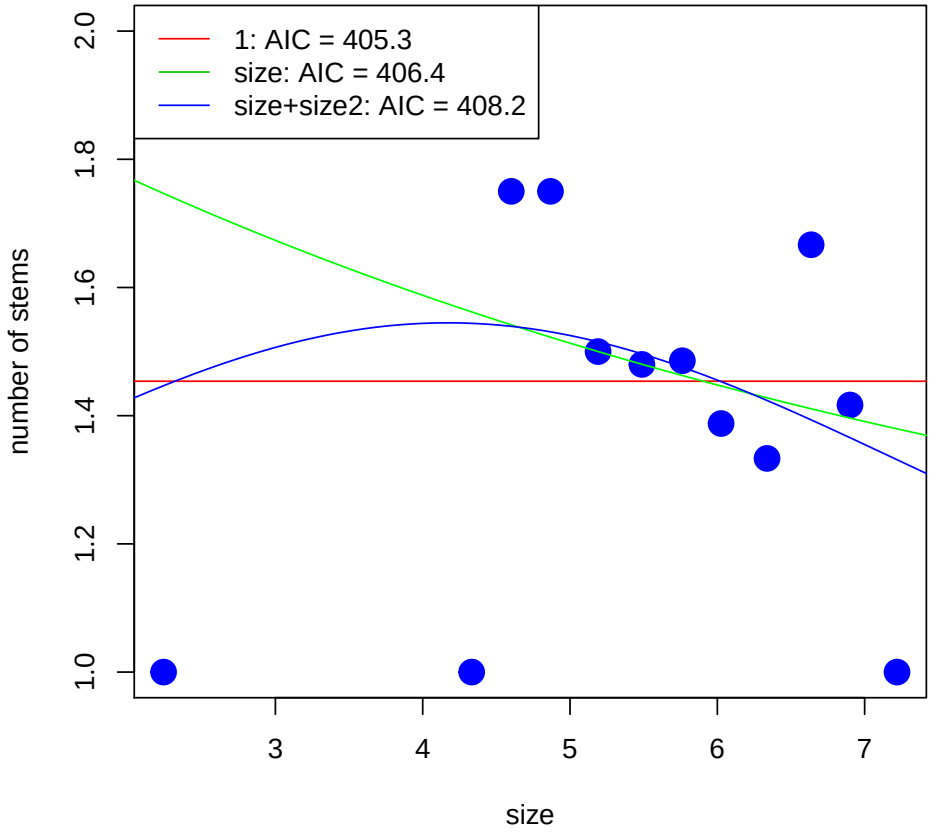

Figure 2.8: Poisson regression models of the number of flowering stems, conditional upon flowering.

Support is lacking for a relation between the number of flowering stems and size, so we proceed with the model containing a constant (intercept term) only: 'fec2Stem $\sim$ 1'.

The third step in the fecundity process is the number of inflorescences (heads) per stem. Here too, the data consist of counts, so we assume a Poisson distribution.

```
fo1<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~1,fec3Head~1),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
fo2<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~1,fec3Head~size),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
fo3<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~1,fec3Head~size+size2),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
```

Notice that because the variable 'fec3Head' contains the average number of flowering heads per stem, warnings appear. This is tolerable, at least for illustration, because these averages are based on the same sample sizes (number of stems). Alternatively, one could have modeled the total count and used the sample size as an offset.

We compare the models of the number of flowering heads per stem:

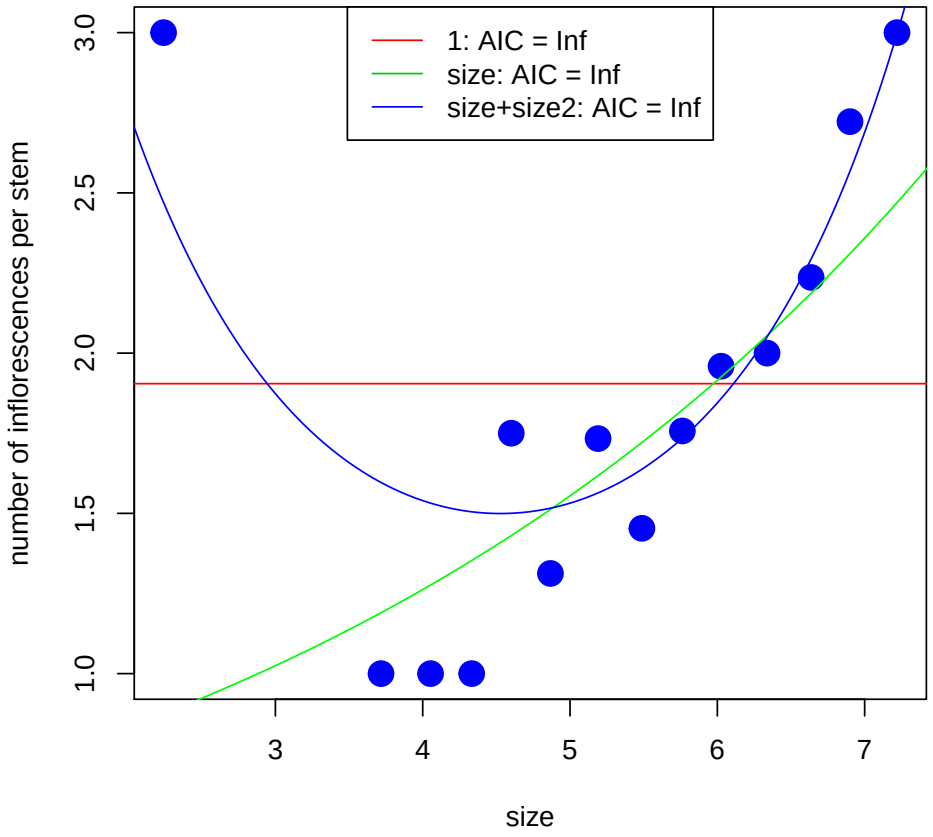

Figure 2.9: Poisson regression models of the number of flowering heads per stem.

Since no AIC values could be calculated for these models, we parsimoniously proceed with the 'fec3Head $\sim$ size' model.

The combined fecundity model is:

```
fo<-makeFecObj(Sp1, Formula=c(fec1Bolt~size+size2,fec2Stem~1,fec3Head~size),
   Family = c("binomial","poisson","poisson"), Transform = c("none","-1","none"),
   fecConstants = data.frame(seedsPerHead=50,seedlingEstablishmentRate= 0.02))
```

where the constants 'seedsPerHead' and 'seedlingEstablishmentRate' specify the average number of seeds per inflorescence and the rate of seedling establishment per seed, respectively, which come from independent data not shown here.

We can now create the F matrix (Fig.2.10). The peak at size=6 indicate that individuals of size 6 contribute most to reproduction, which follows from the unimodal flowering probability in Fig. 2.7.

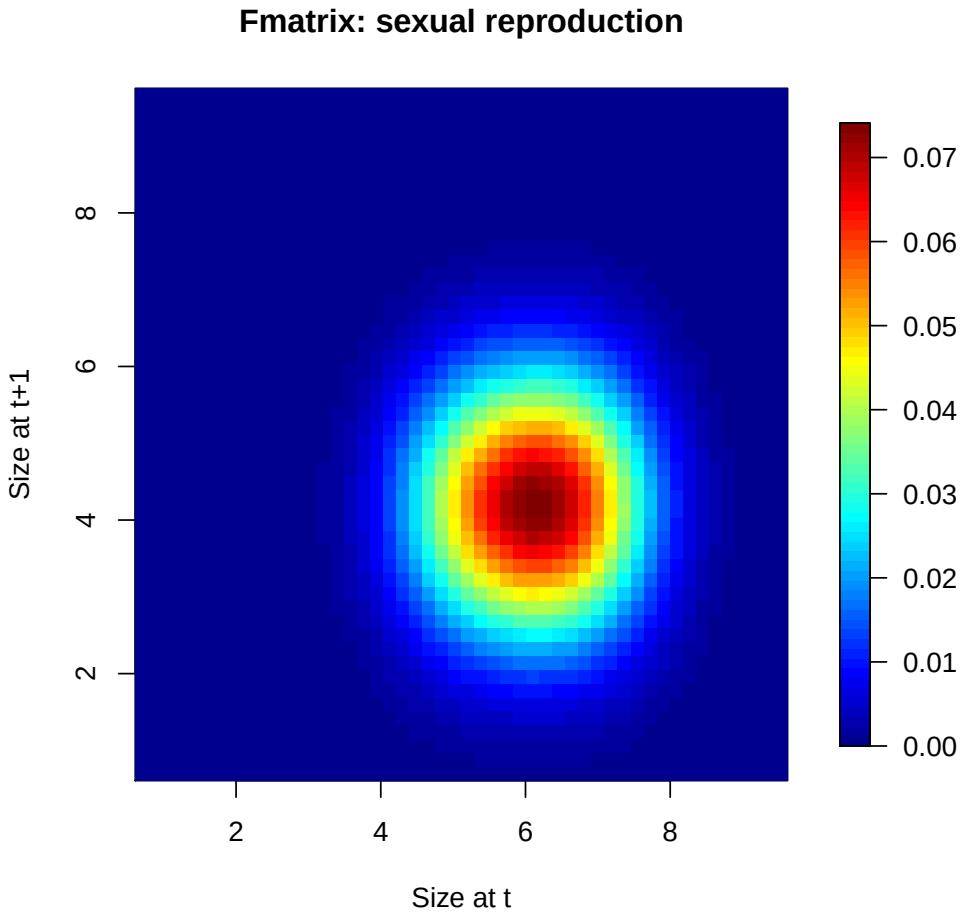

Figure 2.10: F-matrix for *Succisa pratensis*.

2.4.3 Clonality kernel

In addition to survival, growth, and fecundity, *Succisa pratensis* propagates clonally. Axillary buds occasionally form side-rosettes on short stolons; the connection to the parent rosette disintegrates after one year or so. We consider here whether clonal offspring number and size might be related to parent size. As we did with sexual reproduction, we model the process of clone production via a logistic regression (whether or not an individual produces clonal offspring) and a Poisson regression

(of the number of clones). We compare alternative formulations of the logistic regression:

```
co1<-makeClonalObj(Sp1, Formula=cloning~1, Family = c("binomial"),
   Transform=c("none"))
co2<-makeClonalObj(Sp1, Formula=cloning~size, Family = c("binomial"),
   Transform=c("none"))
co3<-makeClonalObj(Sp1, Formula=cloning~size+size2, Family = c("binomial"),
   Transform=c("none"))
```

Plotting the data and models shows no support for a relationship with size (Fig. 2.11):

```
os <- order(Sp1$size)
osClon <- (Sp1$cloning)[os]
osSize<-(Sp1$size)[os]
binnedSize <- tapply(osSize, as.numeric(cut(osSize, breaks=20)),
   mean, na.rm = TRUE); # bin Size data
binnedClon <- tapply(osClon, as.numeric(cut(osSize, breaks=20)),
   mean, na.rm = TRUE) #bin Survival probabilities
plot(binnedSize, binnedClon, pch = 19, cex=2, xlab = "Size at t",
   ylab = "Cloning Probability at t+1",
   main = "Side rosette production")
y0<-predict(co1@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="red")
y0<-predict(co2@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="green")
y0<-predict(co3@fitFec[[1]],newdata=x0,type=response)
lines(x,y0,col="blue")
legend("topright", legend = sprintf("%s: %s = %.1f",c("1","size",
   "size+size2"), c("AIC"), c(AIC(co1@fitFec[[1]]),AIC(co2@fitFec[[1]]),
     AIC(co3@fitFec[[1]]))), col = c(2:4), lty = 1, xjust = 1, bg = "white")
```

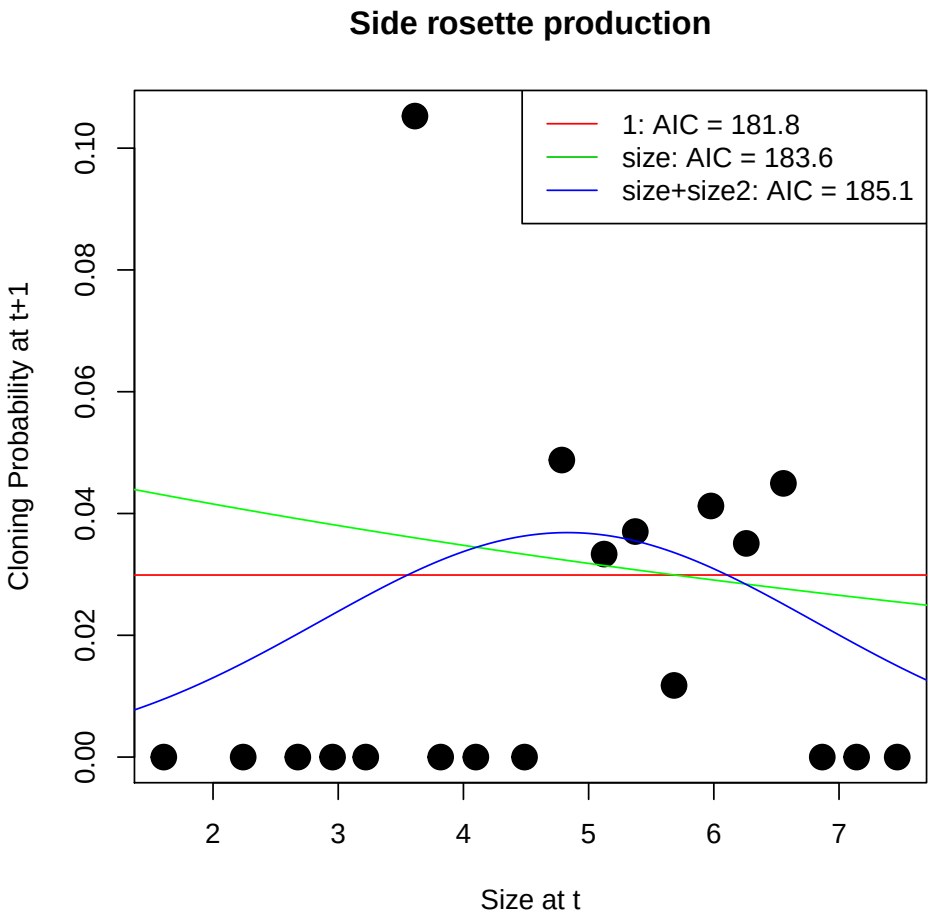

Figure 2.11: Logistic regressions of clone production (side rosettes).

Conditional upon the production of clones, how many clones does a parent rosette produce (Fig. 2.11)?

```
plot(Sp1$size, Sp1$clonesNext, pch = 19, main = "", cex = 2,
   xlab = "Size at t", ylab = "Number of clones at t+1")
```

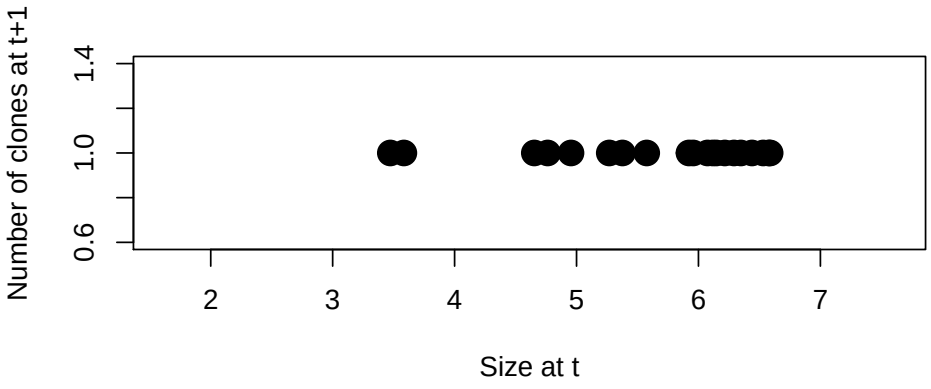

Figure 2.12: Number of offspring rosettes produced by parent rosettes forming at least one offspring rosette.

Clearly, the rate of production of side rosettes is low - a maximum of just one side rosette. We therefore construct the following clonality object:

```
co1 <- makeClonalObj(Sp1, Formula = c(cloning~1, clonesNext~1),
                     Family = c("binomial","poisson"), Transform=c("none","-1"))
```

The last step is to determine the size distribution of the new clonal offspring, and whether offspring rosette size depends on the size of the parent rosette.

```
co1<-makeClonalObj(Sp1, offspringSizeExplanatoryVariables = "1",
                    Formula = c(cloning~1, clonesNext~1),
                    Family = c("binomial","poisson"), Transform=c("none","-1"))
co2<-makeClonalObj(Sp1, offspringSizeExplanatoryVariables = "size",
                    Formula = c(cloning~1, clonesNext~1),
                    Family = c("binomial","poisson"), Transform=c("none","-1"))
co3<-makeClonalObj(Sp1, offspringSizeExplanatoryVariables = "size+size2",
                    Formula = c(cloning~1, clonesNext~1),
                    Family = c("binomial","poisson"), Transform=c("none","-1"))
```

```
plot(Sp1$size[Sp1$offspringNext=="clonal"], Sp1$sizeNext[Sp1$offspringNext=="clonal"],
   pch = 19, main = "", cex = 2,
   xlab = "Parent rosette size at t", ylab = "Offspring rosette size at t+1")
y0<-predict(co1@offspringRel,newdata=x0);lines(x,y0,col="red")
y0<-predict(co2@offspringRel,newdata=x0);lines(x,y0,col="green")
y0<-predict(co3@offspringRel,newdata=x0);lines(x,y0,col="blue")
legend("topleft", col = c(2:4), lty = 1, xjust = 1, bg = "white",
   legend=sprintf("%s: %s = %.1f",c("1","size","size+size2"), c("AIC"),
   c(AIC(co1@offspringRel),AIC(co2@offspringRel),AIC(co3@offspringRel))))
```

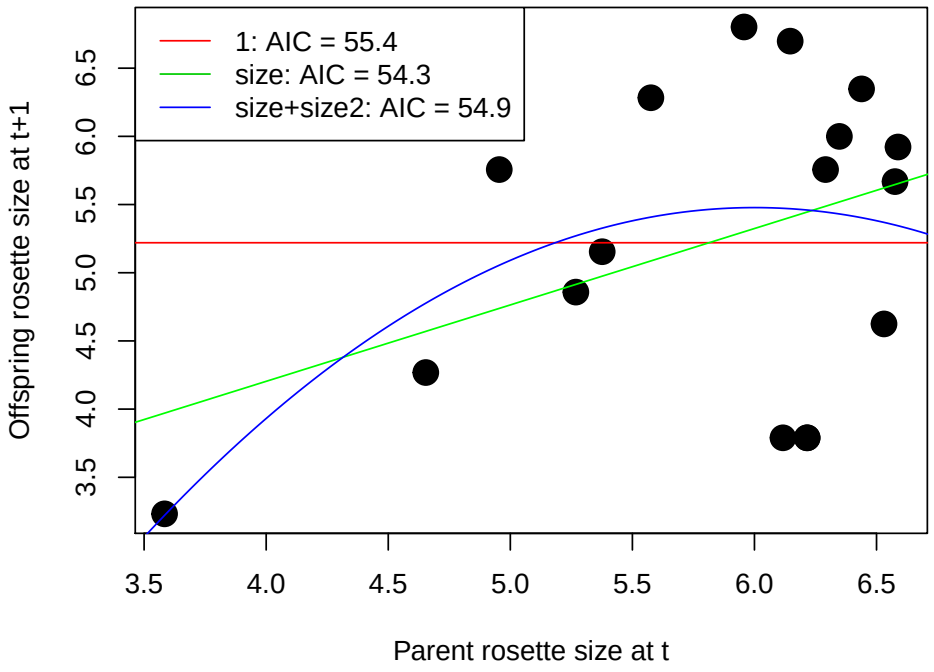

Figure 2.13: Size of side-rosettes regression models.

We choose the linear model for offspring rosette size. Combining these three components together, we create a clonal production object:

```
co<-makeClonalObj(Sp1,
                  offspringSizeExplanatoryVariables = "size",
                  Formula = c(cloning~1, clonesNext~1),
                  Family = c("binomial","poisson"),
                  Transform=c("none","-1"))
Cmatrix<-makeIPMCmatrix(clonalObj=co,minSize=minSize,maxSize=maxSize,
   correction="constant")
```

Note that the offspring size model does not currently use a formula interface, as the other vital rate models do, although this will be updated in a subsequent release of IPMpack. The corresponding C kernel is shown in Fig. 2.14.

```
image.plot(Cmatrix@meshpoints, Cmatrix@meshpoints, t(Cmatrix),
            main = "Cmatrix: clonal propagation",
            xlab = "Size at t", ylab = "Size at t+1")
```

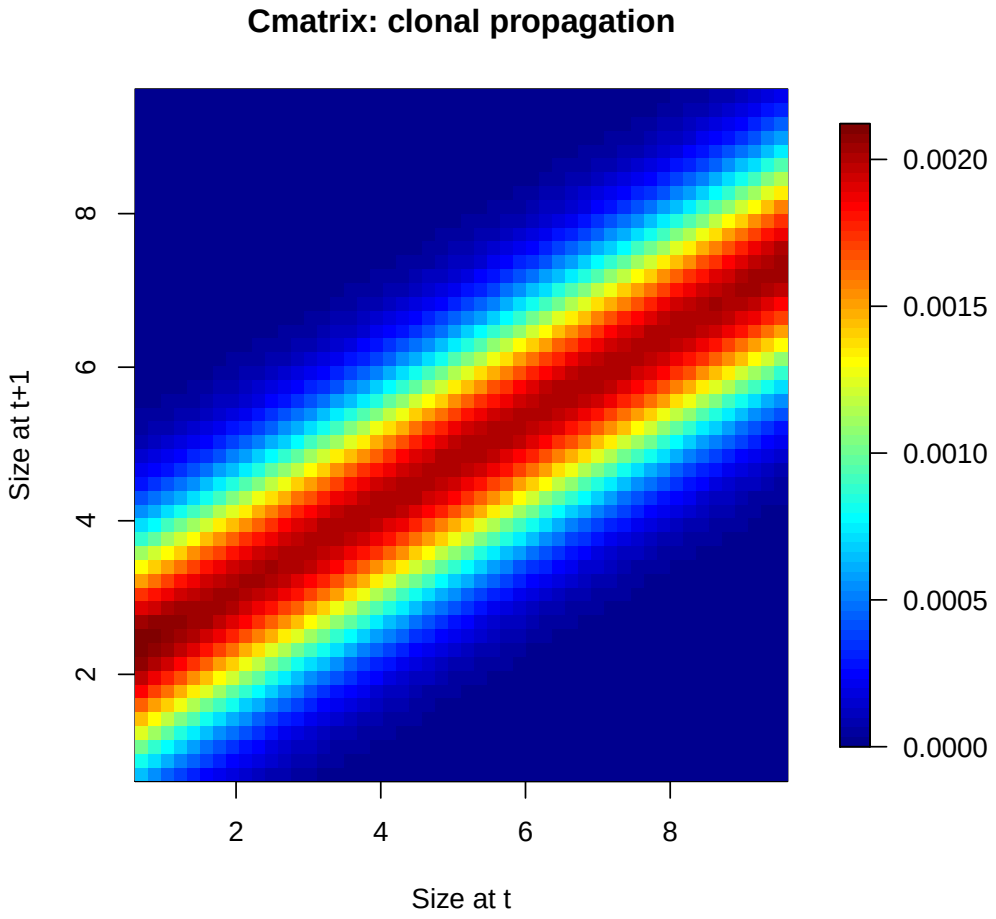

Figure 2.14: Clonality kernel.

2.4.4 Constructing and analyzing the IPM

Now that we have survival/growth, fecundity, and clonality kernels, we simply add them together to produce an Integral Projection Model:

```
IPM <- Pmatrix + Fmatrix + Cmatrix
```

Survival and sexual reproduction dominate the IPM matrix:

```
image.plot(Cmatrix@meshpoints, Cmatrix@meshpoints, t(IPM),
       main = "Pmatrix + Fmatrix + Cmatrix",
       xlab = "Size at t", ylab = "Size at t+1")
```

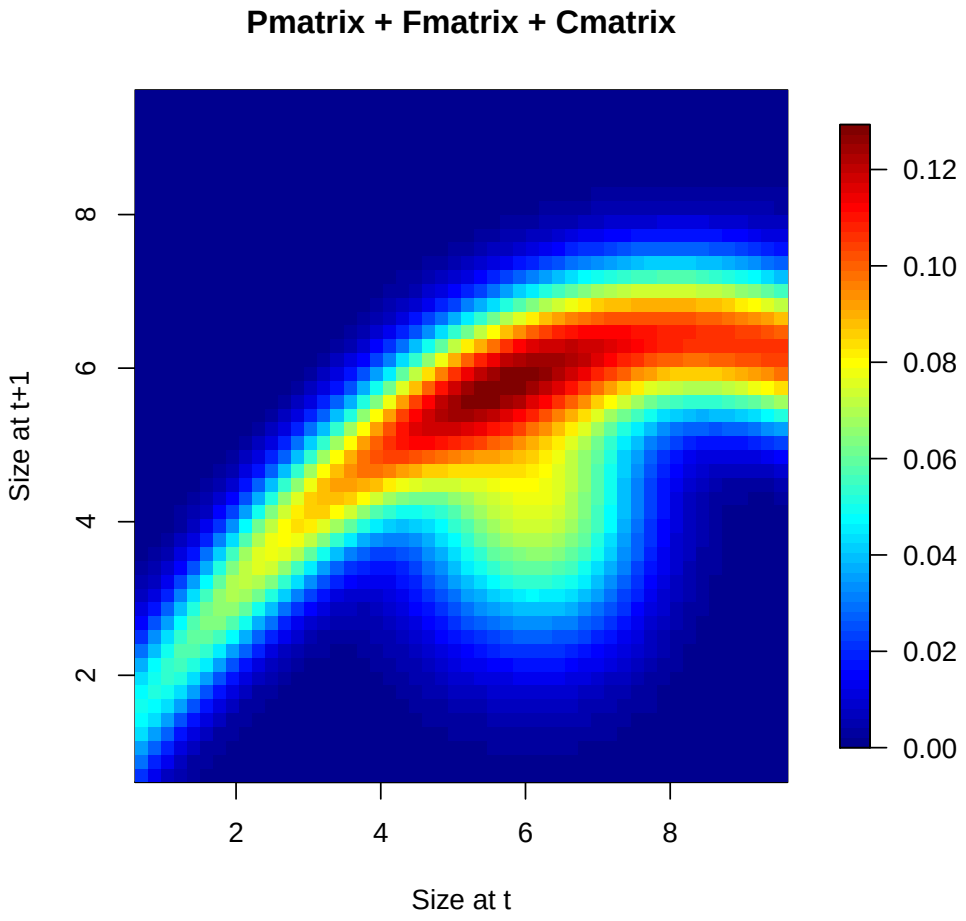

Figure 2.15: IPM for *Succisa pratensis*.

The projected population growth rate (λ) is:

```
(lambda <- Re(eigen(IPM)$value[1]))
```

```
[1] 1.619733
```

Examining the elasticity matrix (de Kroon et al. 2000) of this IPM we can see which parts of the IPM contribute most to λ :

```
elasMatrix<-elas(IPM)
image.plot(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elasMatrix),
           main = "Elasticity", xlab = "Size at t", ylab = "Size at t+1")
```

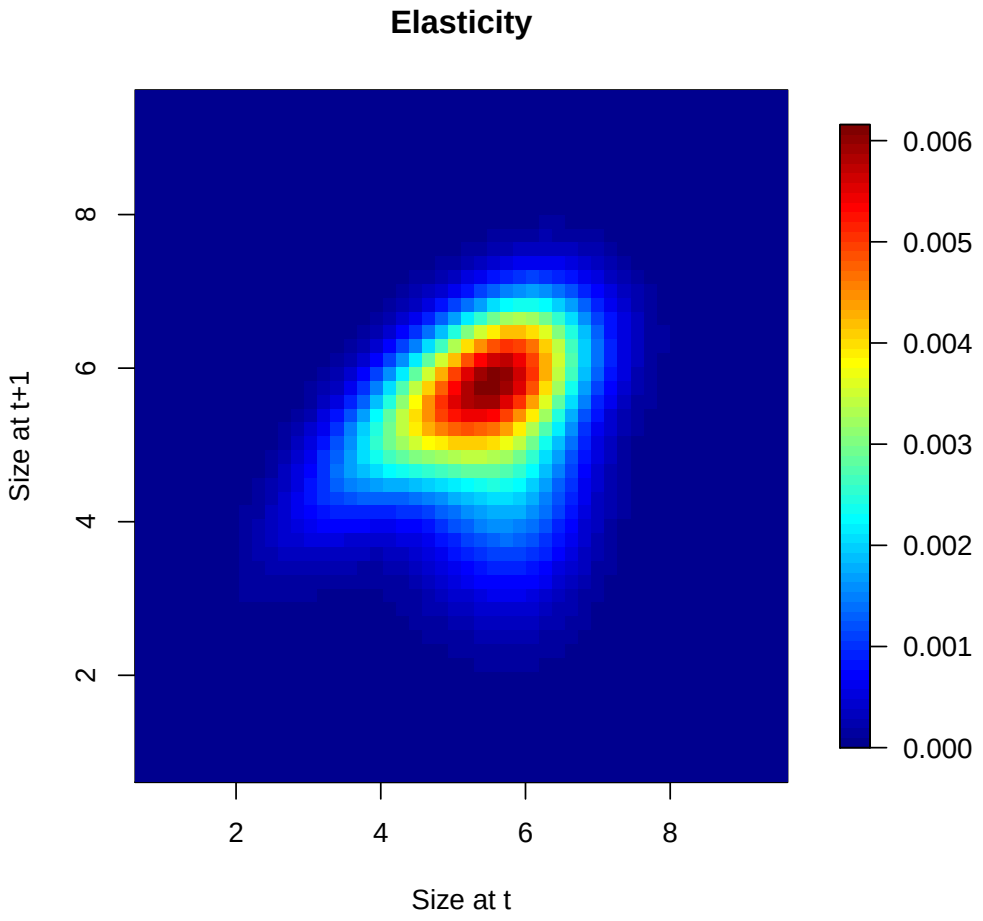

Figure 2.16: λ-Elasticity matrix of the *Succisa pratensis* IPM.

In figure 2.16, we see that individuals of intermediate size have the highest elasticity values. However, it is not clear yet whether this is mostly through survival/growth or through fecundity (or even clonality). Therefore we would like to see the contributions to the elasticity matrix made by the P, F and C matrices underlying the IPM:

```
par(mfrow=c(1,3), mar=c(4.1,4.1,4.1,2.1))
brk<-(0:100)/10000
sensMatrix<-sens(IPM)
elasPMatrix<-Pmatrix*sensMatrix/lambda
image.plot(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elasPMatrix),
           breaks=brk, col=rainbow(100),
           main = "Survival/growth", xlab = "Size at t", ylab = "Size at t+1")
elasFMatrix<-Fmatrix*sensMatrix/lambda
image.plot(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elasFMatrix),
           breaks=brk, col=rainbow(100),
           main = "Fecundity", xlab = "Size at t", ylab = "Size at t+1")
elasCMatrix<-Cmatrix*sensMatrix/lambda
image.plot(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elasCMatrix),
           breaks=brk, col=rainbow(100),
           main = "Clonality", xlab = "Size at t", ylab = "Size at t+1")
```

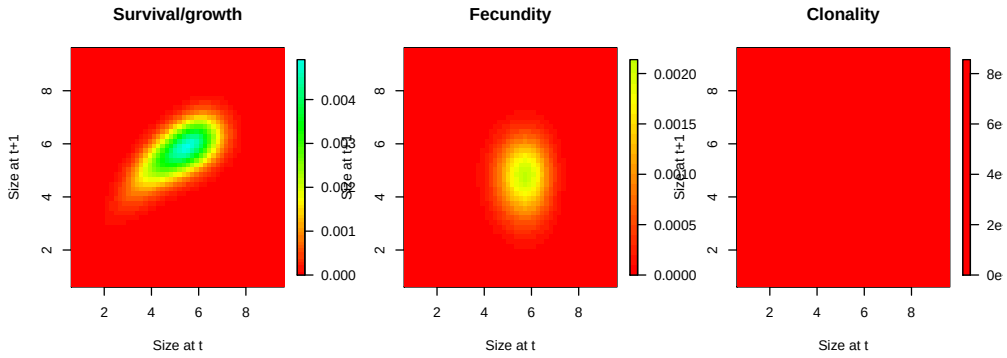

Figure 2.17: λ-Elasticity matrix split into contributions made by the P, F and C matrices.

```
c(sum(elasPMatrix),sum(elasFMatrix),sum(elasCMatrix))
[1] 0.65673525 0.32611530 0.01714945
```

This shows that fecundity contributes 33% to the projected population growth rate (λ), which is surprisingly high for a long-lived species (annual survival of individuals is 0.88, averaged over the size axis: mean(colSums(Pmatrix))). This is probably because of the high λ (1.63) of this IPM - reproduction must be important in the dynamics of a population increasing so rapidly.

As we saw in figure 2.16, intermediately-sized individuals are most important for population growth. This might be because only few individuals reach larger sizes. Large individuals of *Succisa pratensis* do survive at a higher rate and produce larger side-rosettes, but figure 2.7 showed optimal bolting probabilities for medium-sized individuals, and very low bolting probabilities for the largest individuals. That is very surprising and could be an alternative explanation for the high elasticity values of intermediately-sized individuals. To figure out what has caused the shown elasticity pattern, and the surprising peak in flowering probability near size 6, we should consider an alternative state variable, since the currently used state variable ('size') only includes rosette leaves but does not account for stem leaves Fig. 2.18.

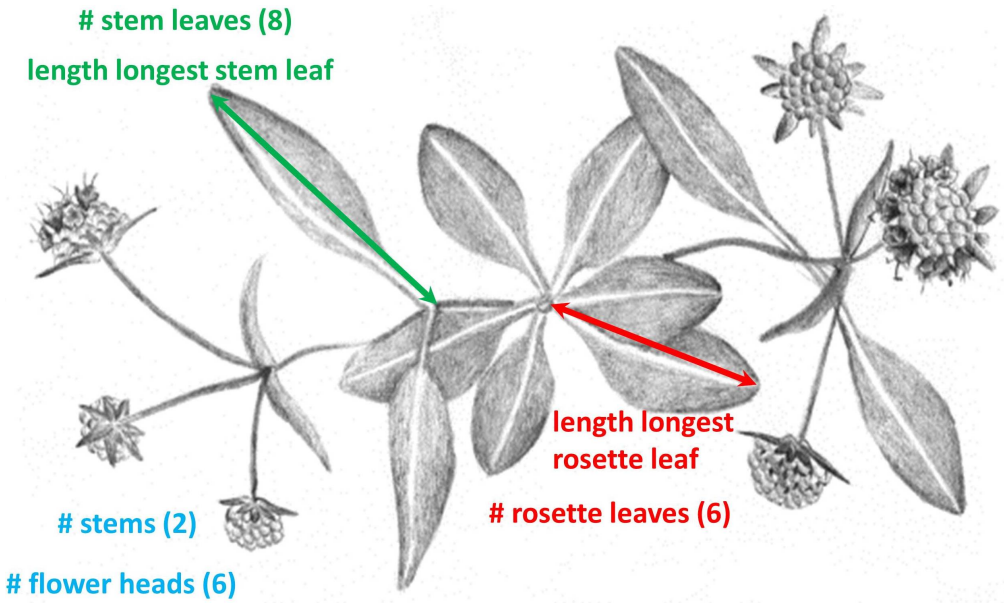

Figure 2.18: Rosette and stem leaves of *Succisa pratensis*. Drawing by Lidewij H. Keser, from Jongejans (2004).

2.4.5 Reanalysis with a different state variable

We investigate the problem by rebuilding the IPM for *Succisa pratensis* with a different state variable: approximate total leaf area.

First we have to load a new data set. In this new data set, 'size' is defined as the log of the sum of the products of leaf number and maximum leaf length for rosette and stem leaves separately. This data set also is available in *IPMpack* (from version 2.0 onward):

```
data(dataIPMpackSuccisa2)
Sp2 <- dataIPMpackSuccisa2
```

We define a new size axis for the midpoint evaluation of the IPMs:

```
x<-seq(from=0,to=10,length=1001)
x0<-data.frame(size=x,size2=x*x)
minSize<-min(Sp2$size,na.rm=T)-1
maxSize<-max(Sp2$size,na.rm=T)+2
```

We redo the survival analysis:

```
survModelComp(dataf = Sp2,
              makePlot = TRUE,
              legendPos = "bottomright",
              mainTitle = "Survival")
```

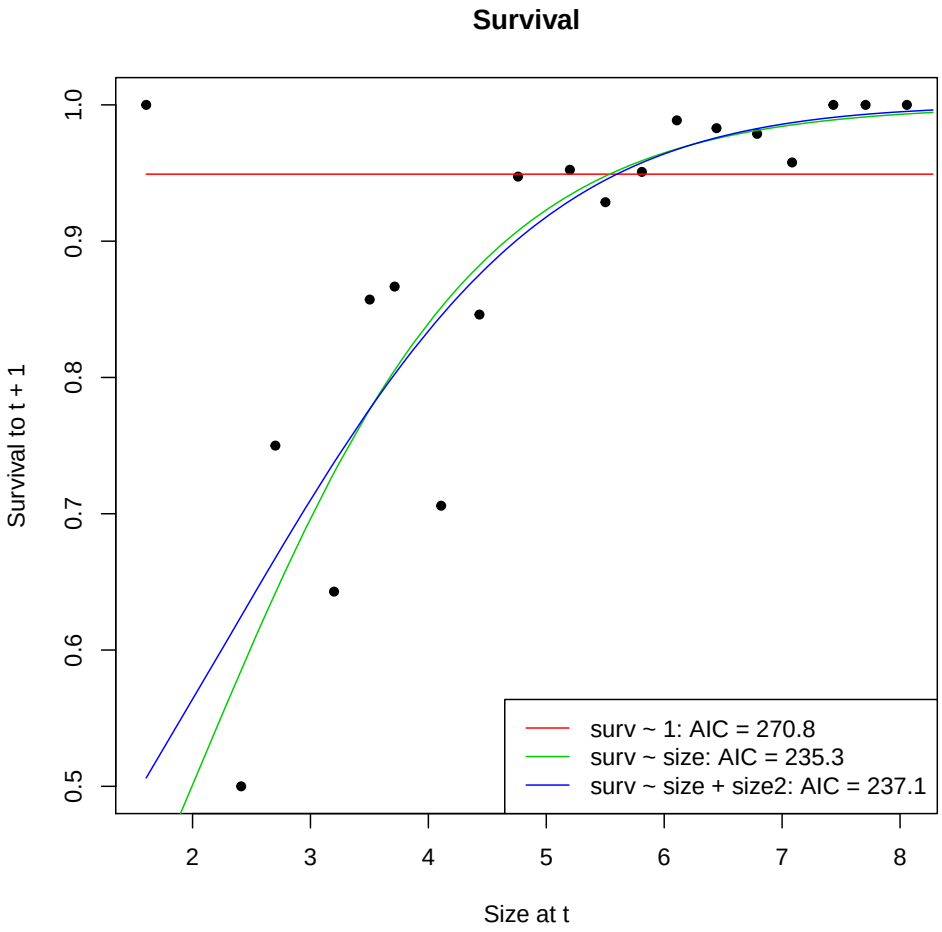

Figure 2.19: Survival model comparison for a new state variable.

With the new size variable, survival depends more clearly on size, which is encouraging.

```
(so<-makeSurvObj(Sp2,surv~size))
An object of class "survObj"
Slot "fit":

Call:  glm(formula = Formula, family = binomial, data = dataf)

Coefficients:
(Intercept)         size
    -1.6490       0.8264
```

```
Degrees of Freedom: 668 Total (i.e. Null);  667 Residual
Null Deviance:            268.8
Residual Deviance: 231.3        AIC: 235.3
```

Next, we perform a new analysis of growth:

```
growthModelComp(dataf = Sp2,
                makePlot = TRUE,
                legendPos = "bottomright",
                mainTitle = "Growth")
```

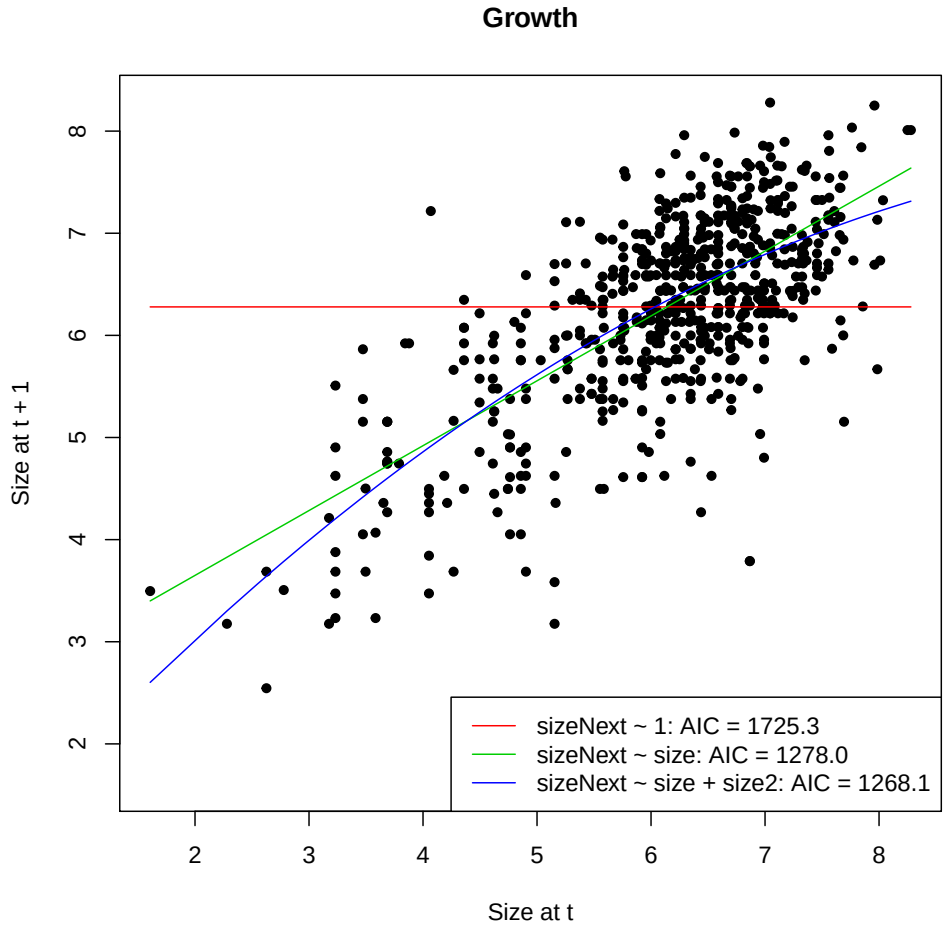

Figure 2.20: Growth model comparison for a new state variable.

Now, the best-fit model involves a quadratic function of size.

```
(go<-makeGrowthObj(Sp2,sizeNext~size+size2))
An object of class "growthObj"
Slot "fit":
```

```
Call:
lm(formula = Formula, data = dataf)

Coefficients:
(Intercept)         size        size2
    0.73070      1.25385     -0.05541


Slot "sd":
[1] 0.6541827
```

From these survival and growth objects we can create a P matrix (Fig. 2.21).

```
Pmatrix2<-makeIPMPmatrix(survObj=so, growObj=go, minSize=minSize, maxSize=maxSize,
                         correction="constant")
image.plot(Pmatrix2@meshpoints, Pmatrix2@meshpoints, t(Pmatrix2),
           main = "Pmatrix: survival and growth",xlab = "Size at t",
           ylab = "Size at t+1")
```

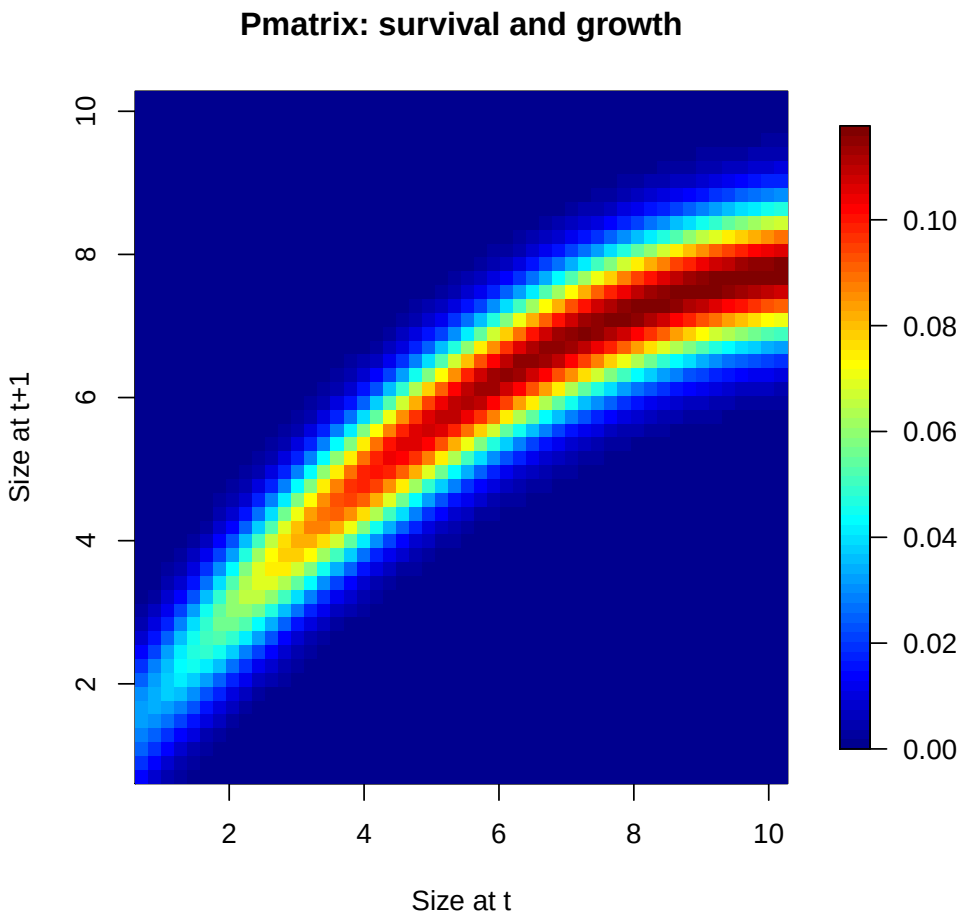

Figure 2.21: Survival/growth kernel (P) for *Succisa pratensis*, using the new state variable.

The kernel exhibits the same qualitative pattern of growth and survival as the previous model in Fig.2.5.

We redo the fecundity analyses, starting with the probability of flowering (bolting):

```
fo1<-makeFecObj(Sp2, Formula=fec1Bolt~1, Family = "binomial")
fo2<-makeFecObj(Sp2, Formula=fec1Bolt~size, Family = "binomial")
fo3<-makeFecObj(Sp2, Formula=fec1Bolt~size+size2, Family = "binomial")
```

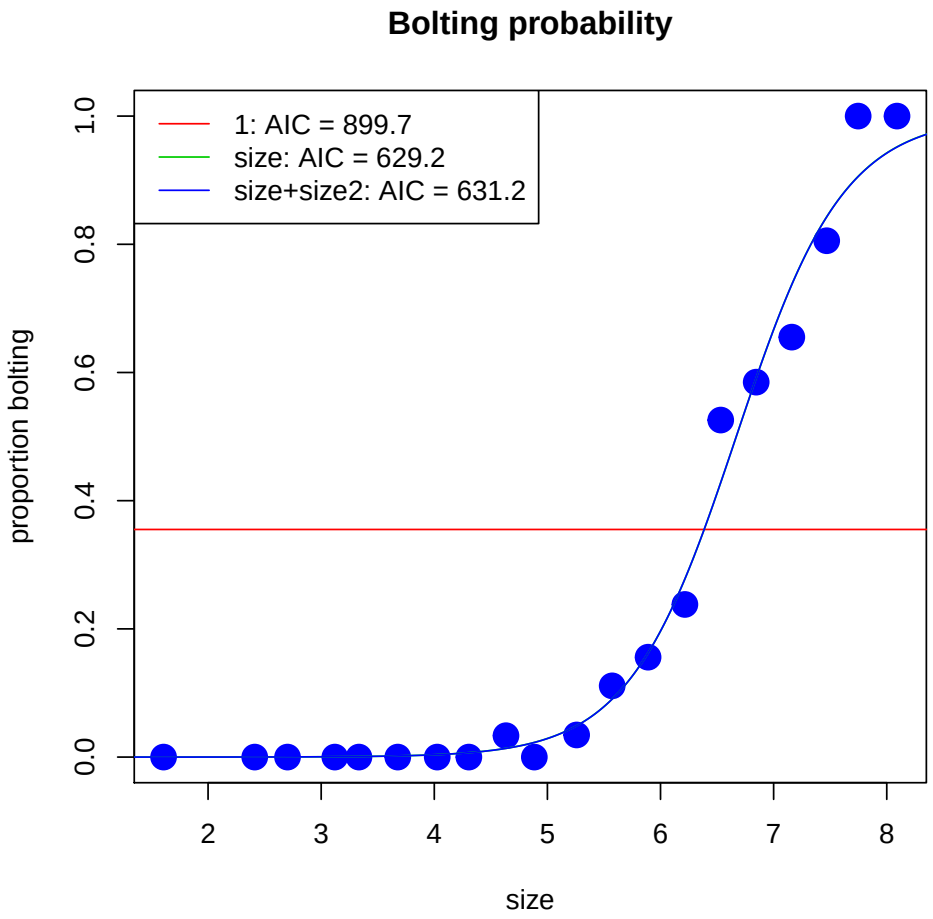

Figure 2.22: Logistic regressions of the probability of flowering, using the new state variable.

Figure 2.22 shows a less surprising pattern: flowering probability increases with size (Fig. 2.22). Next, we look at the number of stems produced by flowering plants (*i.e.*, those that produce at least one flowering stem):

```
fo1<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~1),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
fo2<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~size),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
fo3<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~size+size2),
                Family = c("binomial","poisson"), Transform = c("none","-1"))
```

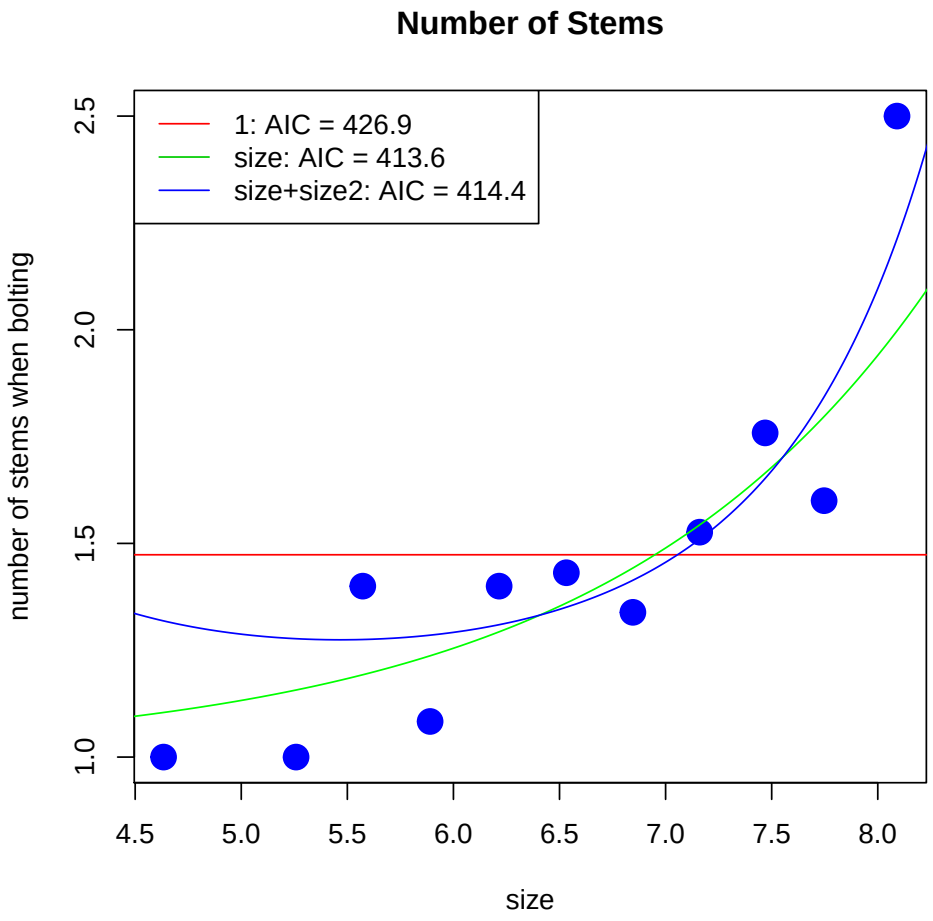

Figure 2.23: Poisson regressions of the number of stems, using the new state variable.

Inspection of figure 2.23 leads us to select the 'fec2Stem ∼ size' model. The next step is a reanalysis of the number of flower heads per stem:

```
fo1<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~size,fec3Head~1),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
fo2<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~size,fec3Head~size),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
fo3<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,fec2Stem~size,fec3Head~size+size2),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"))
```

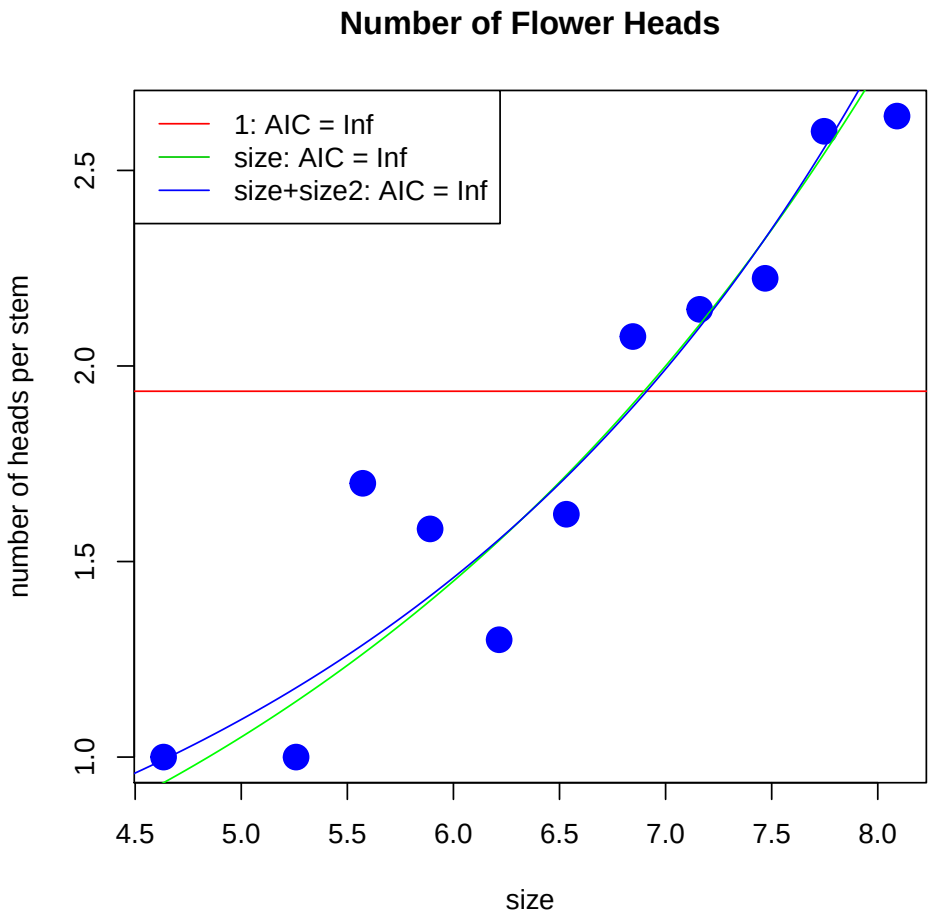

Figure 2.24: Poisson regressions of the number of inflorescences per stem, using the new state variable.

For parsimony, we select the 'fec3Head ~ size' model. We can now produce an F matrix (Fig. 2.25):

```
fo <- makeFecObj(Sp2, Formula=c(fec1Bolt~size,
        fec2Stem~size, fec3Head~size),
                Family = c("binomial","poisson","poisson"),
                Transform = c("none","-1","none"),
                fecConstants = data.frame(seedsPerHead=50,
                             seedlingEstablishmentRate= 0.02))
```

```
Fmatrix2<-makeIPMFmatrix(fecObj=fo, minSize=minSize, maxSize=maxSize,
                         correction="constant")
image.plot(Fmatrix2@meshpoints, Fmatrix2@meshpoints, t(Fmatrix2),
           main = "Fmatrix: sexual reproduction",xlab = "Size at t",
           ylab = "Size at t+1")
```

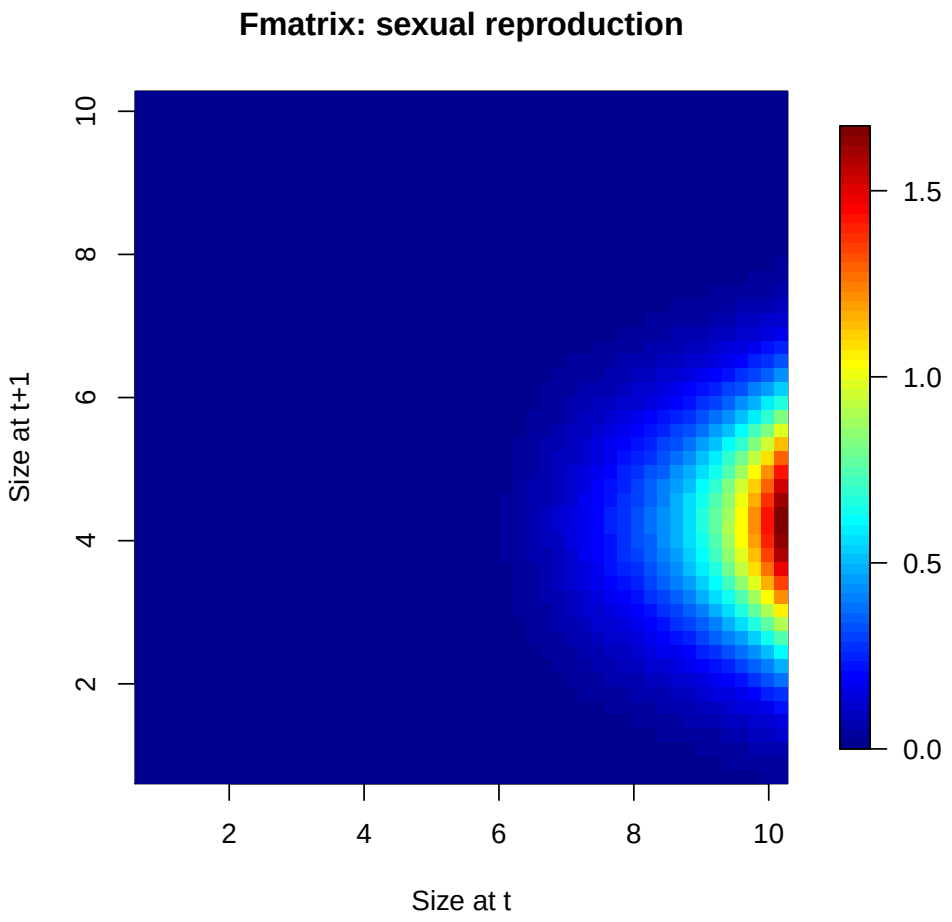

Figure 2.25: F matrix of the sexual reproduction kernel, using the new state variable.

We also have to redo the analyses of clonal propagation, and construct a new C matrix, which results in figure 2.26:

```
co<-makeClonalObj(Sp2, offspringSizeExplanatoryVariables = "size",
                  Formula = c(cloning~1, clonesNext~1),
                  Family = c("binomial","poisson"),
                  Transform=c("none","-1"))
Cmatrix2<-makeIPMCmatrix(clonalObj=co,
                         minSize=minSize,
                         maxSize=maxSize,
                         correction="constant")
image.plot(Cmatrix2@meshpoints, Cmatrix2@meshpoints, t(Cmatrix2),
           main = "Cmatrix: clonal propagation",
           xlab = "Size at t", ylab = "Size at t+1")
```

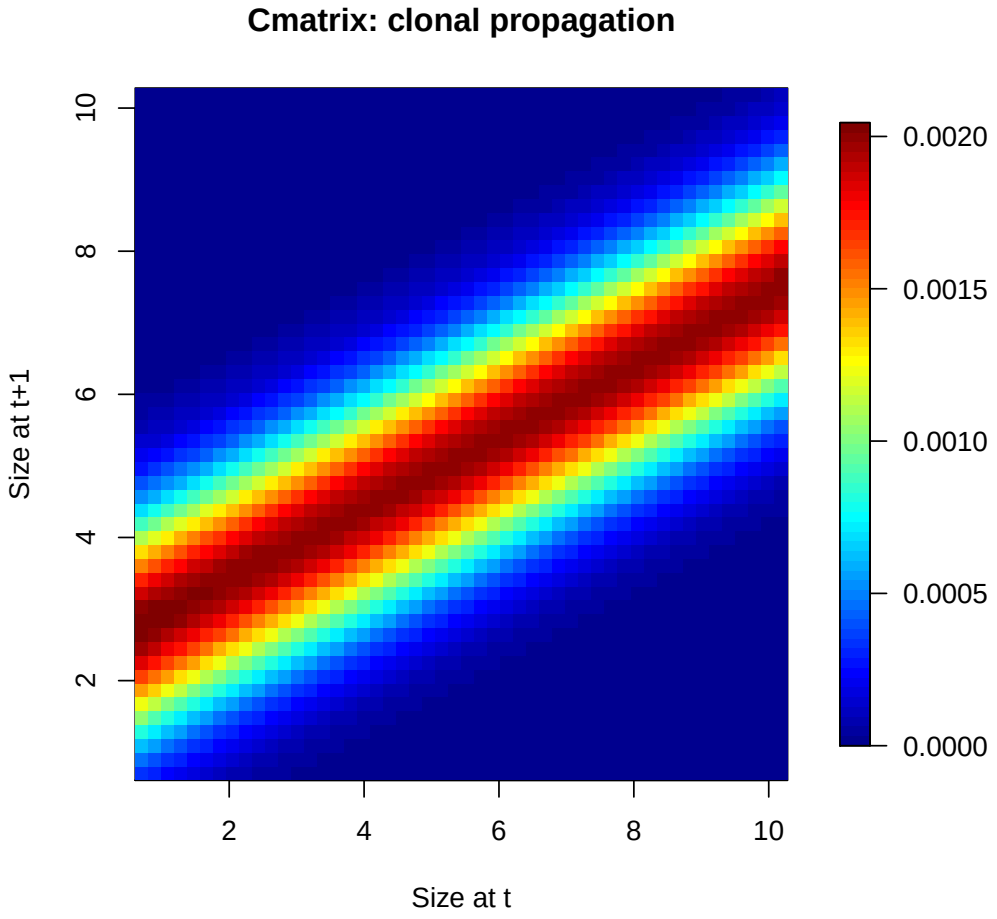

Figure 2.26: C matrix of the clonal propagation kernel, using the new state variable.

We construct a new IPM, based on the new state variable:

```
IPM2 <- Pmatrix2 + Fmatrix2 + Cmatrix2
round(Re(eigen(IPM2)$value[1]),2)
[1] 1.45
```

Now we investigate which vital rates the projected population growth rate is most sensitive to (Figs. 2.27 and 2.28). The function *sensParams* calculates for each model parameter the first-order derivative with respect to the projected population growth rates. The function calculates this numerically by changing each parameter in turn a little bit, and then obtaining the resulting population growth rate projection. The ratio of the change in population growth rate and the change in the model parameter gives an estimation of the population growth rate to change in the model parameters.

```
sensitivityOutput<-sensParams(growObj=go, survObj=so, fecObj=fo, clonalObj=co,
                              nBigMatrix=50, minSize=minSize, maxSize=maxSize)
barplot(sensitivityOutput$elas,
        main = expression("Parameter elasticity of population growth rate "* lambda),
        las = 2, cex.names = 0.5)
```

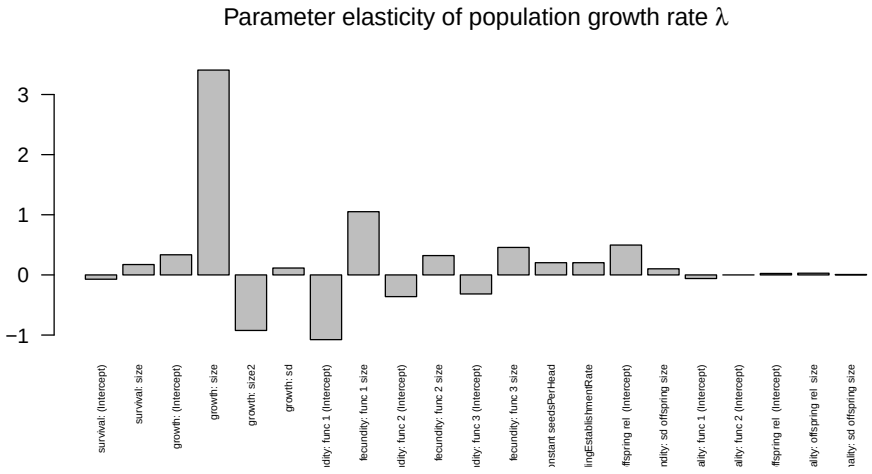

Figure 2.27: Parameter elasticity.

```
elasMatrix2<-elas(IPM2)
image.plot(Pmatrix@meshpoints, Pmatrix@meshpoints, t(elasMatrix2),
           main = "Elasticity", xlab = "Size in t", ylab = "Size in t+1")
```

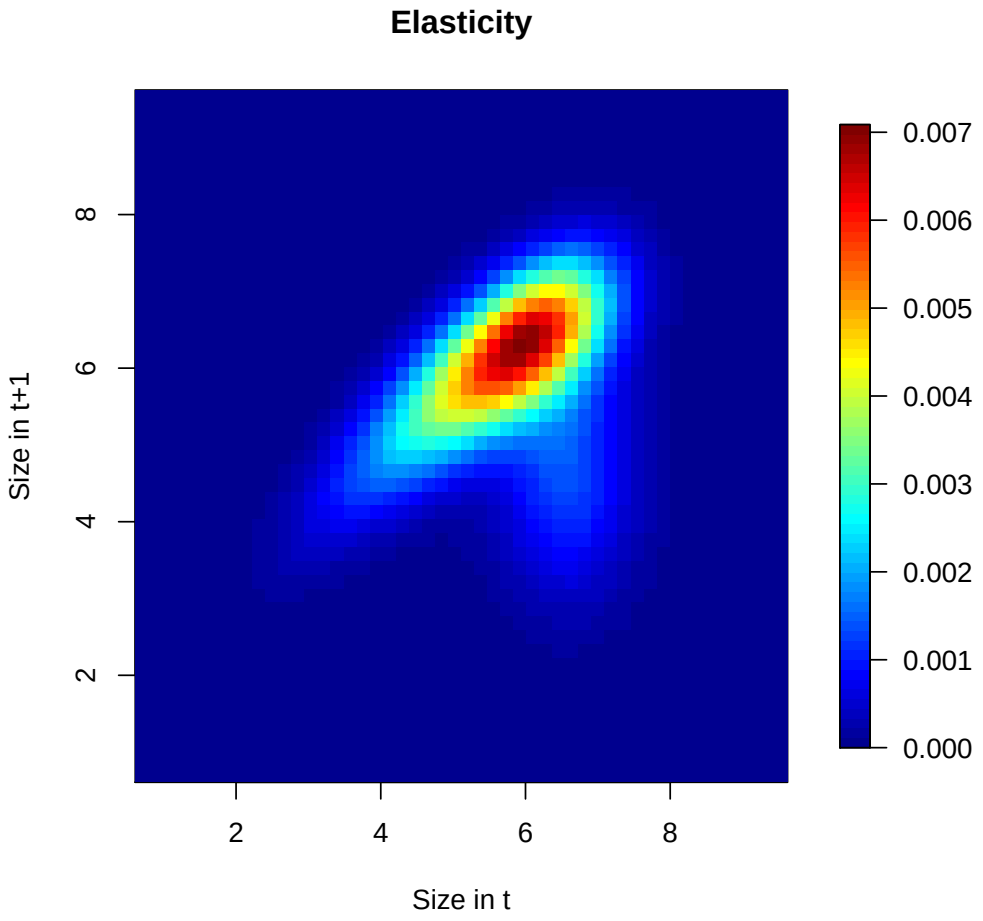

Figure 2.28: Elasticity matrix of the new IPM.

We see that intermediate size still contributes most to population growth (FIG.2.28), but this second IPM better meets our expectations about the biology of the species, as well as fitting the data better, compared to the first.

2.4.6 Alternative models with fewer components in the reproductive pathway

So far, we have made the F matrix with three reproductive steps: flowering probability, number of stems when flowering, and number of heads per stem. The final exercise is to build simpler models for reproduction and investigate the implications of model complexity for model output. We compare an IPM with two reproductive steps (flowering (y/n), and number of flower heads when flowering) vs. an IPM with one reproductive step (number of flower heads per individual, whether they are flowering (heads>0) or not (heads=0)).

First we calculate the number of flower heads per flowering plant for the two-step model of reproduction (flowering probability(x) * number of flower heads (x)) and analyse this new variable:

```
Sp2$totalHeads <- Sp2$fec2Stem * Sp2$fec3Head
fo1<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,totalHeads~1),
                Family = c("binomial","poisson"), Transform = c("none","none"))
fo2<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,totalHeads~size),
                Family = c("binomial","poisson"), Transform = c("none","none"))
fo3<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,totalHeads~size+size2),
                Family = c("binomial","poisson"), Transform = c("none","none"))
```

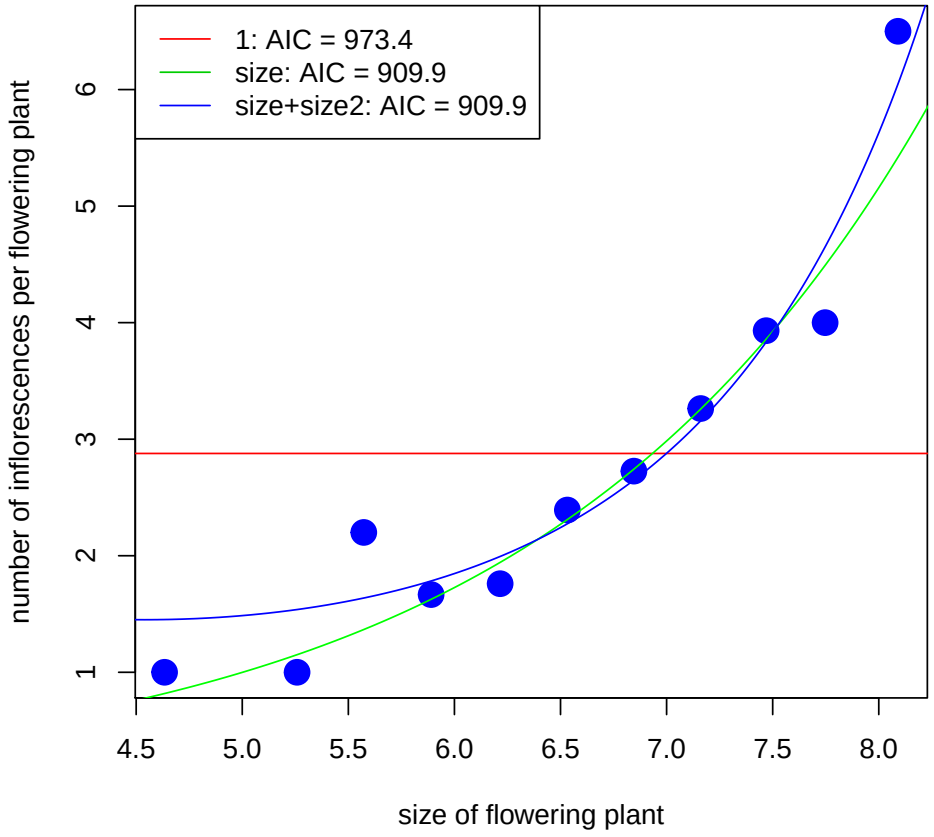

Figure 2.29: Models of the total number of heads per flowering plant.

It is a bit of a toss-up between the 'totalHeads~size' and 'totalHeads~size+size2' models in Figure 2.29, so we will implement each in turn. First we rebuild the IPM with 'totalHeads~size':

```
fo<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,totalHeads~size),
```

```
        Family=c("binomial","poisson"), Transform=c("none","none"),
        fecConstants=data.frame(seedsPerHead=50,seedlingEstablishmentRate=0.02))
 Fmatrix3<-makeIPMFmatrix(fecObj=fo, minSize=minSize,
                          maxSize=maxSize, correction="constant")
 IPM3 <- Pmatrix2 + Fmatrix3 + Cmatrix2
 round(Re(eigen(IPM3)$value[1]),2)
[1] 1.44
```

The resulting projected population growth rate is almost the same as that of IPM2. Using two instead of three regressions to make the fecundity kernel, with exactly the same data, yields a very similar population growth rate, but we potentially learn more about reproduction if we divide the process into more steps. Rebuilding the IPM with 'totalHeads∼size+size2' does not change λ:

```
 fo<-makeFecObj(Sp2, Formula=c(fec1Bolt~size,totalHeads~size+size2),
        Family = c("binomial","poisson"),
        Transform = c("none","none"),
        fecConstants = data.frame(seedsPerHead=50, seedlingEstablishmentRate=0.02))
 Fmatrix4<-makeIPMFmatrix(fecObj=fo,minSize=minSize,maxSize=maxSize,
    correction="constant")
 IPM4 <- Pmatrix2 + Fmatrix4 + Cmatrix2
 round(Re(eigen(IPM4)$value[1]),2)
[1] 1.45
```

Nonetheless, we complete the exercise by simplifying the IPM further, modelling sexual reproduction as a one-step process. First we calculate the number of flower heads for all plants (reproductive or not) using the 'number of flower heads (x)' model. Non-flowering plants have zero inflorescences.

```
 Sp2$fecundity <- Sp2$fec2Stem * Sp2$fec3Head
 Sp2$fecundity[Sp2$fec1Bolt==0] <- 0
 fo1<-makeFecObj(Sp2, Formula=c(fecundity~1),
                 Family = c("poisson"), Transform = c("none"))
 fo2<-makeFecObj(Sp2, Formula=c(fecundity~size),
                 Family = c("poisson"), Transform = c("none"))
 fo3<-makeFecObj(Sp2, Formula=c(fecundity~size+size2),
                 Family = c("poisson"), Transform = c("none"))
```

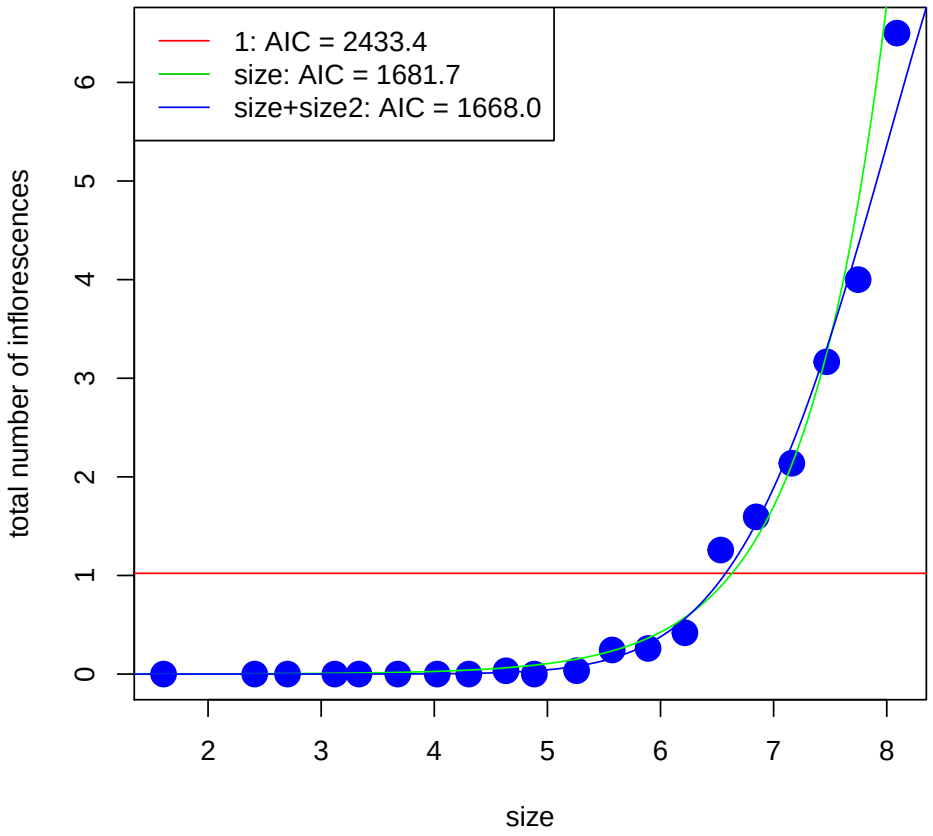

Figure 2.30: One-step model of fecundity.

Fitting various models to this new, one-step fecundity model, we see in figure 2.30 that the 'fecundity∼size+size2' model has the lowest AIC. For the sake of comparison, we build IPMs with the different fecundity models. First we build an IPM with constant fecundity with respect to parent size: 'fecundity∼1':

```
fo<-makeFecObj(Sp2, Formula=c(fecundity~1),
                Family = c("poisson"), Transform = c("none"),
                fecConstants = data.frame(seedsPerHead=50,
                seedlingEstablishmentRate=0.02))
Fmatrix5<-makeIPMFmatrix(fecObj=fo,minSize=minSize,maxSize=maxSize,
   correction="constant")
IPM5 <- Pmatrix2 + Fmatrix5 + Cmatrix2
round(Re(eigen(IPM5)$value[1]),2)
[1] 1.92
```

This λ is much higher, probably because the fecundity of small plants (e.g. seedlings) is overestimated. Next we build an IPM with 'fecundity∼size':

```
fo<-makeFecObj(Sp2, Formula=c(fecundity~size),
                Family = c("poisson"), Transform = c("none"),
                fecConstants = data.frame(seedsPerHead=50,
                seedlingEstablishmentRate=0.02))
Fmatrix6<-makeIPMFmatrix(fecObj=fo,minSize=minSize,maxSize=maxSize,
   correction="constant")
IPM6 <- Pmatrix2 + Fmatrix6 + Cmatrix2
round(Re(eigen(IPM6)$value[1]),2)
[1] 1.47
```

Then we use the 'fecundity∼size+size2' model:

```
fo<-makeFecObj(Sp2, Formula=c(fecundity~size+size2),
                Family = c("poisson"), Transform = c("none"),
                fecConstants = data.frame(seedsPerHead=50, seedlingEstablishmentRate=0.02))
Fmatrix7<-makeIPMFmatrix(fecObj=fo,minSize=minSize,maxSize=maxSize,
   correction="constant")
IPM7 <- Pmatrix2 + Fmatrix7 + Cmatrix2
round(Re(eigen(IPM7)$value[1]),2)
[1] 1.45
```

We see that the last two IPMs result in similar asymptotic population growth rates, compared to the IPMs with more complex models for reproduction. There are, of course, other reasons to prefer more complex models of reproduction. One problem with the one-step reproduction model is that it suffers from zero-inflation (deviates from a Poisson distribution by having too many zero-valued data).

2.5 Conclusions

In this appendix we have illustrated the construction of Integral Projection Models using the R package *IPMpack*, focusing particularly on examples of how to model sexual reproduction, and the comparison of fecundity models of varying complexity. Based on the same data, the projected population growth rate did not change when we varied the number of steps in the reproductive pathway from 3 to 2 to 1. However, including more steps in the fecundity kernel allows for a more in-depth analysis (e.g. parameter sensitivity in Fig. 2.27) of which vital rates contribute most to (variation in) population growth and other model output parameters.

We also demonstrated how to model clonal production, otherwise known as asexual reproduction. Further, we illustrated the consequences of different state variables: in our case study it did matter whether 'size' was based on rosette leaves only, or on both rosette and stem leaves. Since flowering rosettes produce relatively few rosette leaves (compared to non-flowering rosettes

of similar size the year before), ignoring stem leaves resulted in low flowering probabilities for large individuals. This shows that the choice of the state variable is very important and that it is good practice to study the robustness of one's analyses by exploring alternative state variables. In this case different state variables also resulted in rather different population growth rate projections.

For further complexities in fecundity models, a number of other published studies can provide guidance on decision making during model construction: multiple discrete states: Appendix F; Hesse et al. 2008; Williams et al. 2010; seed bank: Appendix C; Salguero-Gomez et al. 2012).

2.6 References

- Adams, A.W. (1955) *Succisa pratensis* Moench. (*Scabiosa succisa* L.). Journal of Ecology, 43, 709-718.
- de Kroon, H., van Groenendael, J.M. & Ehrlen, J. (2000). Elasticities: a review of methods and model limitations. Ecology, 81, 607-618.
- Hesse, E., Rees, M. & Muller Scharer, H. (2008). Life-History Variation in Contrasting Habitats: Flowering Decisions in a Clonal Perennial Herb (Veratrum album). The American Naturalist, 172, E196-E213.
- Hooftman, D.A.P. & Diemer, M. (2002). Effects of small habitat size and isolation on the population structure of common wetland species. Plant Biology, 4, 720-728.
- Jongejans, E. (2004). Life history strategies and biomass allocation. The population dynamics of perennial plants in a regional perspective. PhD thesis. Wageningen University.
- Jongejans, E. & de Kroon. H. (2005). Space versus time variation in the population dynamics of three co-occurring perennial herbs. Journal of Ecology, 93, 681-692.
- Metcalf, C., McMahon, S.M., Salguero-Gomez, R. & Jongejans, E. (2013). IPMpack: an R package for integral projection models. Methods in Ecology and Evolution, 4, 195-200.
- Salguero-Gomez, R., Siewert, W., Casper, B. B., & Tielberger, K. (2012). A demographic approach to study effects of climate change in desert plants. Philosophical Transactions of the Royal Society B: Biological Sciences, 367, 3100-3114.
- Schaminee, J.H.J., Stortelder, A.H.F. & Weeda, E.J. (1996). De vegetatie van Nederland. Deel 3. Plantengemeenschappen van graslanden, zomen en droge heiden. Opulus Press, Uppsala.
- Williams, J., Auge, H. & Maron, J. (2010). Testing hypotheses for exotic plant success: parallel experiments in the native and introduced ranges. Ecology, 91, 1355-1366.

Chapter 3

Appendix C: IPMs for Complex Life Cycles

Contact: Roberto Salguero-Gomez (Salguero@demogr.mpg.de)

Abstract

This appendix shows how to build an IPM for the plant *Hypericum cumulicola* (Hypericaceae) including both continuous and discrete stages. We use a subset of the long-term census data collected and described by Quintana-Ascencio et al. (2003) to estimate vital rates for two transitions, 1994-1995, and 1996-1997. We then estimate and investigate differences in population growth rates λ via a Life Table Response Experiment *(LTRE)* analysis.

3.1 Introduction

In this appendix, the user will learn how to incorporate discrete stages into an IPM using the R package *IPMpack* (Metcalf et al. 2013). Discrete stages might include seed banks, seedlings (Appendix F), or dormant stages (Jacquemyn et al. 2010) in plants or instars in insects. First we describe the life cycle of our study species, *Hypericum cumulicola*, then we construct two IPMs describing the population dynamics from year t to year $t+1$ for data collected in 1994-1995 and in 1996-1997.

We extract basic demographic output from each IPM, then we explore the effect each vital rate has on the differences between estimated population growth rates in 1994-1995 vs. 1996-1997s through a Life Table Response Experiment (LTRE; Caswell 2001). A LTRE offers a more mechanistic understanding of how and why population growth rates (or any population statistics from which sensitivities can be calculated) differ as a function of their underlying processes (*e.g.*, survival, growth, probability of flowering, probability of seed germination, etc).

3.2 Study system/objectives

The primary goal of this appendix is to illustrate how to construct IPMs where the life cycle of the species is based not only on a continuous stage variable (*e.g.*, volume, size, height), but also one or more discrete stage variables (*e.g.*, seed bank, dormant individuals, diseased individuals, etc). We use data on the population dynamics of *Hypericum cumulicola* (Hypericaceae). Other valuable examples of how to create models with discrete data can be found within *IPMpack* , using the random data generator *generateData(type="discrete")*.

Hypericum cumulicola is an endangered, fire-dependent, perennial herb endemic to Florida scrub. Individuals of this species are iteroparous and short-lived. Their seeds can remain viable in the soil longer than one year. Thus, populations consist of vegetative individuals (plants) as well as seeds in a soil seed bank. The latter stage will be treated here as a discrete stage (see Eager et al. 2013 for a treatment of seed banks as a continuous stage using soil depth in an IPM framework) More information about the biology and demography of the species can be found in Quintana-Ascencio, Menges & Weekley (2003).

We model the life-cycle of *H. cumulicola* using an annual time-step, as described in Metcalf et al. (2013). Briefly, photosynthetically active individuals were sampled every May at Archbold Biological Station, Florida (USA). The size of each individual was measured as its stem maximum height. Likewise, the number of fruits produced per individual were counted. The number of seeds contained in each fruit, as well as the number of seeds that germinate the same year of their production vs. enter, survive in, or emerge from the soil seed bank were estimated from additional data as described in Quintana-Ascencio et al. (2003).

3.3 Data

We begin by reading in data. The data file is part of the R package *IPMpack* (from version 2.0 onward):

```
require(IPMpack)
data(dataIPMpackHypericumCov)
d <- dataIPMpackHypericumCov
```

Calling the help manual of *dataIPMpackHypericumCov* will display a brief description of the data and the various variables in it:

```
help(dataIPMpackHypericumCov)
head(d)
```

```
  id bald fireYear initial year TSLF size ontogeny fec0
1  5    1     1967    1994 1994   27    1        1    0
2 22    1     1967    1994 1994   27    1        1    0
3 33    1     1967    1994 1994   27    1        1    0
4 41    1     1967    1994 1994   27    2        1    0
5 55    1     1967    1994 1994   27    2        1    0
6 42    1     1967    1994 1994   27    2        1    0
  fec1 fec2     fec3      fec4       goSB staySB      cov
1   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
2   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
3   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
4   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
5   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
6   NA   NA 0.001216 0.1414214 0.08234362  0.672 1481.836
  surv sizeNext ontogenyNext  covNext
1    1        6            0 1420.622
2    0       NA           NA 1420.622
3    0       NA           NA 1420.622
4    1        3            0 1420.622
5    1       11            0 1420.622
6    1       13            0 1420.622
```

This data set contains the variables *size* and *sizeNext*, measured as the maximum stem height (cm) of individuals in the population *bald 1* at times t and $t+1$, respectively, as well as the variable *surv*, *i.e.*, whether individuals survived (=1) or not (=0) from time t to $t+1$. This information would be sufficient to explore passage time (see *passageTime()* in *IPMpack*) and age from stage decompositions (see *sizeToAge()*) as described by Caswell (2001) for above ground individuals. But here we will model the full life cycle of the species. To that end, the data set also contains additional variables: the probability of being reproductive, *feco0*, the number of fruits produced per individual, *fec1*, and the number of seeds per fruit, *fec2*. Columns *fec3* and *fec4* concern the

probability of germination and the probability of seedling survival within the year of seed and seedling production, respectively, while *goSB* and *staySB* are constants describing the dynamics of the seed bank that are explained below.

```
d1 <- subset(d,is.na(d$size)==FALSE | d$ontogenyNext==1)
```

For the purposes of this appendix, we will analyze only a subset of the data for the first two annual periods: 1994-1995, and 1996-1997. We know *a priori* that these years were very different in terms of precipitation (P. Quintana-Ascencio, pers. comm.), but otherwise the choice is arbitrary; users are encouraged to explore building IPMs and running the consequent LTREs with the other years available in this data set.

```
d1 <- subset(d1,d1$year==1994 | d1$year==1996)
```

Additional experiments carried out by Quintana-Ascencio and Menges (pers. comm.) provided estimates of the following vital rates, which we apply to all the studied annual transitions. These constants are easiest to store outside of the *d1* data frame:

- Number of seeds produced per fruit:

  ```
  fec2 <- 13.78
  ```

- Probability of germination within the year of seed production:

  ```
  fec3 <- 0.001336
  ```

- Probability of seedling survival from the time of germination to the time of the next annual census *t+1*, corresponding to approximately 6 months:

  ```
  fec4 <- 0.14
  ```

- Probability of a seed entering the seed bank:

  ```
  goSB <- 0.08234528
  ```

- Probability of a seed staying in the seed bank:

  ```
  staySB <- 0.671
  ```

Next, we re-organize the data, eliminating some information that will not be used here.

```
d1 <- d1[,c("year","size","surv","sizeNext","fec0","fec1")]
head(d1)
```

```
  year size surv sizeNext fec0 fec1
1 1994    1    1        6    0   NA
2 1994    1    0       NA    0   NA
3 1994    1    0       NA    0   NA
4 1994    2    1        3    0   NA
5 1994    2    1       11    0   NA
6 1994    2    1       13    0   NA
```

```
tail(d1)
```

```
    year size surv sizeNext fec0 fec1
296 1996   NA   NA       11   NA   NA
297 1996   NA   NA       11   NA   NA
298 1996   NA   NA       11   NA   NA
299 1996   NA   NA       12   NA   NA
300 1996   NA   NA       12   NA   NA
303 1996   NA   NA       14   NA   NA
```

In order to understand how the data should be organized for analysis in *IPMpack*, we will take a look at individuals in rows *51* through *55* below. Note that individuals that survive from time t to time $t+1$ (*surv = 1*) will have data for both the variables *size* and *sizeNext*, as is the case for the individuals in rows *51* through *53*. In contrast, an individual that dies (*surv = 0*) will have *sizeNext = NA*, as is the case for the individual in row *55*. If the data are not set-up in this way, *IPMpack* will automatically generate NA values in the variable *sizeNext* for those individuals where *surv = 0*. This produces a warning message, so the user is aware of the generation of NA data. The same rationale applies for fecundity data: if an individual at time t is not reproductive (*fec0 = 0*, as in individual *52* in this example, the fecundity variables that are conditional on *fec0* must be NA (*fec1 = NA* here). Note that an individual's probability of reproduction *fec0* does not need to be conditional on their survival, and vice versa, unless there are biological constraints on the species (*e.g.*, masting, costs of reproduction, etc).

```
d1[51:55,]
   year size surv sizeNext fec0 fec1
51 1994   40    1       17    1   41
52 1994   40    1       37    0   NA
53 1994   40    1       48    1   39
54 1994   42    0       NA    1  182
55 1994   43    0       NA    1  369
```

In preparing data where more than one state variable will be considered, the user must specify which transitions and contributions are made to the continuous stage (changes in maximum stem length in this example), and which ones are made into and out of discrete stages (*e.g.*, seed bank). The following lines of code set the variable *stage* (maximum height of individuals of *Hypericum*) as continuous in the IPM. If a given individual has not yet been recruited into the continuous stage (or sampled), its *size* will be *NA*, and thus its *stage* will also be *NA*. If an individual has not survived (*surv = 0*), its *stageNext* will be *dead*.

```
d1$stageNext <- d1$stage <- "continuous"
d1$stage[is.na(d1$size)] <- NA
d1$stageNext[d1$surv==0] <- "dead"
head(d1)
  year size surv sizeNext fec0 fec1      stage  stageNext
1 1994    1    1        6    0   NA continuous continuous
```

```
2 1994    1    0       NA    0   NA continuous       dead
3 1994    1    0       NA    0   NA continuous       dead
4 1994    2    1        3    0   NA continuous continuous
5 1994    2    1       11    0   NA continuous continuous
6 1994    2    1       13    0   NA continuous continuous
```

Next we specify the number of individuals changing from a specific size at time t to another (or the same) value at time *t+1*. Because individuals are very unlikely to have the exact same data for size transitions (*e.g.*, from 2 cm in 1995 to 3 cm in 1996), we simply assign a "1" to the variable *number* for every row. This variable will come in quite handy below, when merging this (continuous) part of the life cycle with the discrete transitions.

```
d1$number <- 1
head(d1)
```

```
  year size surv sizeNext fec0 fec1      stage  stageNext
1 1994    1    1        6    0   NA continuous continuous
2 1994    1    0       NA    0   NA continuous       dead
3 1994    1    0       NA    0   NA continuous       dead
4 1994    2    1        3    0   NA continuous continuous
5 1994    2    1       11    0   NA continuous continuous
6 1994    2    1       13    0   NA continuous continuous
  number
1      1
2      1
3      1
4      1
5      1
6      1
```

Next we specify the transitions within and among the continuous and discrete stages - transitions from the continuous stage to the discrete stage, transitions within the discrete stage, and transitions from the discrete stage to the continuous stage. In the case of *Hypericum* these transitions are:

- Continuous stage -> Discrete stage: Individuals with a given height in year t contribute seeds to the seed bank, that is, seeds that were produced, did not germinate, and remain viable.
- Discrete stage -> Discrete stage: Prolonged dormancy and survival of seeds in the seed bank.
- Discrete stage -> Continuous stage: Germination of seeds from the seed bank to become individuals of a given above ground stem height.

To accommodate these possible transitions, we create a data frame that will describe these processes. Here is where the variable *number* in *d1* comes in handy: there will be hundreds of transitions from the discrete stage to the discrete stage, and it would be rather burdensome to

store a row in the data frame for each seed. The variable *d1$number* will indicate how many individuals are making each type of transition; *IPMpack* will automatically read those numbers and consider them in the growth and fecundity objects (below).

```
seedbank <- data.frame(year="All",size=NA,surv=1,sizeNext=NA,fec0=NA,fec1=NA,
                       stage=c("seedbank","seedbank","continuous"),
                       stageNext=c("seedbank","continuous","seedbank"),
                       number=c(staySB,(1-staySB)*fec3*fec4,1))
```

```
seedbank
```

```
  year size surv sizeNext fec0 fec1      stage  stageNext
1  All   NA    1       NA   NA   NA   seedbank   seedbank
2  All   NA    1       NA   NA   NA   seedbank continuous
3  All   NA    1       NA   NA   NA continuous   seedbank
        number
1 6.710000e-01
2 6.153616e-05
3 1.000000e+00
```

The data for continuous and discrete parts of the lifecycle are then concatenated into a single data set. Note that we have specified *year = "All"* for the seed bank dynamics because, as the user will see below, processes governing these dynamics are assumed to remain constant across years.

```
d1 <- rbind(d1,seedbank)
```

```
head(d1)
```

```
  year size surv sizeNext fec0 fec1      stage  stageNext
1 1994    1    1        6    0   NA continuous continuous
2 1994    1    0       NA    0   NA continuous       dead
3 1994    1    0       NA    0   NA continuous       dead
4 1994    2    1        3    0   NA continuous continuous
5 1994    2    1       11    0   NA continuous continuous
6 1994    2    1       13    0   NA continuous continuous
  number
1      1
2      1
3      1
4      1
5      1
6      1
```

```
tail(d1)
```

```
     year size surv sizeNext fec0 fec1      stage
299  1996   NA   NA       12   NA   NA       <NA>
300  1996   NA   NA       12   NA   NA       <NA>
```

```
303  1996   NA   NA       14   NA   NA       <NA>
2181  All   NA    1       NA   NA   NA   seedbank
2191  All   NA    1       NA   NA   NA   seedbank
2201  All   NA    1       NA   NA   NA continuous
      stageNext       number
299  continuous 1.000000e+00
300  continuous 1.000000e+00
303  continuous 1.000000e+00
2181   seedbank 6.710000e-01
2191 continuous 6.153616e-05
2201   seedbank 1.000000e+00
```

The final step before data exploration and analysis is to specify *stage* and *stageNext* as factors:

```
d1$stage <- as.factor(d1$stage)
d1$stageNext <- as.factor(d1$stageNext)
```

3.4 Analysis

Having gained a good understanding of the organization of the data and the different vital rates involved in the life cycle of *Hypericum cumulicola*, in the next section we construct the IPM models. But first, it is necessary to have a good grasp on the integral equations that will govern the IPMs. The first equation below describes how the size distribution of individuals in time t changes to a size distribution of individuals in time t as a function of the kernel $\boldsymbol{K}$. This kernel contains two very different demographic processes: changes in size/stage conditional on survival, described by the sub-kernel $\boldsymbol{P}$, and the per-capita contribution of reproductive individuals to new recruits, described by the sub-kernel $\boldsymbol{F}$.

$$n_{t+1}(z') = \int_{\Omega} K(z) n_t(z) dz = \int_{\Omega} [P(z',z) + F(z',z)] n_t(z) dz \tag{3.1}$$

The sub-kernel $\boldsymbol{P}$ is further decomposed into two vital rates: the probability that individuals of within a given stage (*i.e.* continuous or discrete) survive as a function of their size x in time *(*t), and the probability that, having survived in that time interval, an individual of a size x in time t will grow, stay the same size, or shrink to size y in time *t+1*.

$$P(z',z) = surv(z)\ growth(z',z) \tag{3.2}$$

Likewise, the sub-kernel $\boldsymbol{F}$ is a function of a series of conditional demographic processes, or vital rates. As it is typically the case in the $\boldsymbol{F}$, more vital rates will be modeled than in the sub-kernel $\boldsymbol{P}$ due to the slightly more complicated nature of reproduction in comparison to survival/changes in size of already established individuals. The processes to consider here are:

- *fec0* - probability of reproduction

- *fec1* - conditional on reproduction, the number of fruits produced
- *fec2* - number of seeds per fruit
- *fec3* - probability that seeds will germinate within the year of their production
- *fec4* - probability of seedling from the time of germination to the time of the next annual census
- *staySB* - probability that seeds in the seed bank will survive a time interval and remain in the seed bank
- *goSB* - probability that seeds emerge as seedlings
- *(*fec5) - size distribution of germinated individuals

More information on each of these processes is provided in the next section. These are combined into the fecundity kernel below

$$F(z',z) = fec0(z)\ fec1(z)\ fec2\ (1-goSB)\ fec3\ fec4\ fec5(z')\ + \\ fec0(z)\ fec1(z)\ fec2\ goSB\ (1-staySB)\ fec3\ fec4\ fec5(z')$$

To build the model, we will first create, step by step, the IPM for the 1994-1995 transition, illustrating the criteria for model selection. Then we will repeat the same procedure for the 1996-1997 transition, in a more expedited fashion. Users are encouraged to repeat the model-building and model comparison steps detailed for the 1994-1995 IPM with the 1996-1997 data to explore model selection, as well with all other years available in this data set.

```
d94 <- subset(d1, d1$year == "1994" | d1$year == "All")
```

We first make a dummy size axis and define the size range. This size axis will be used to define the mesh points for integration of the IPM kernel, following the midpoint rule.

```
minSize<-min(d1$size,na.rm=T)
maxSize<-max(d1$size,na.rm=T)
x<-seq(from=minSize,to=maxSize,length=1001)
x0<-data.frame(size=x,size2=x^2,size3=x^3) # for later use
```

3.4.1 Survival and growth kernel

We begin building the IPM kernel with a survival analysis. We use the 'survModelComp' function to explore whether survival is a function of size in a polynomial manner from zero to three degrees, yielding figure 3.1:

Based on this visual exploration of the data, accompanied by AIC scores, we select the linear model with a quadratic term for the survival model. At this point, it is worth cautioning against choosing statistical models of vital rates purely on the basis of AIC scores. A massive body of

```
survModelComp(dataf = d94[!is.na(d$size),],
      expVars = c(surv~1, surv~size, surv~size + size2,
      surv~size + size2 + size3),
      makePlot = TRUE, legendPos = "bottomleft", mainTitle = "Survival")
```

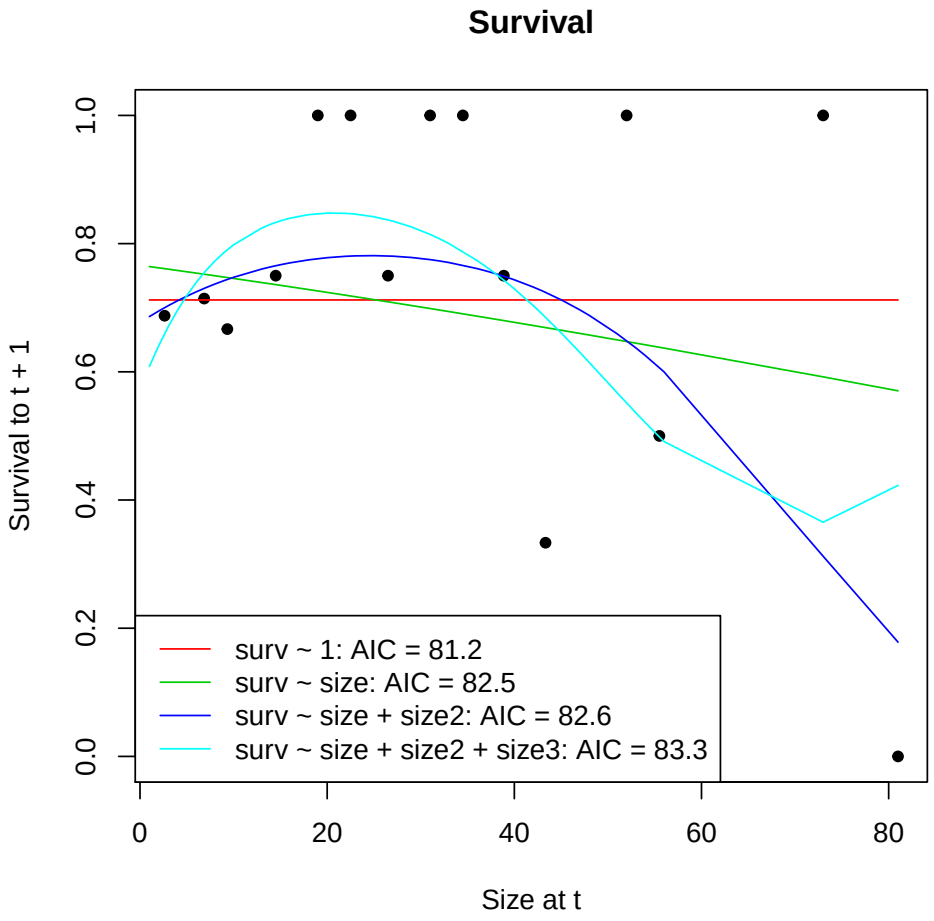

Figure 3.1: Survival of *Hypericum cumulicola* individuals from 1994-95 (points).

literature has been devoted to model selection, and this is not the place to discuss it. However, mathematical modeling should always be guided by biological knowledge of the system to be modeled, and not the other way around. In this case, consulting with the experts who collected the data confirmed that larger individuals have a high probability of mortality, and that a regression with a quadratic term plausibly describes this phenomenon:

```
so94 <- makeSurvObj(d94, surv~size+I(size^2))
```

The next step in building the IPM kernel is an analysis of growth, *i.e.*, the change in size of individuals who survived between 1994 and 1995. We use the following *IPMpack* function to make figure 3.2. The dashed, gray line is the 1:1 ratio of sizes at times *t+1* and *t*. Naturally, points above that 1:1 line are individuals that survived between 1994 and 1995 and grew, whereas individuals below that diagonal became smaller from 1994 to 1995.

```
growthModelComp(dataf = d94, expVars = c(sizeNext~1, sizeNext~size,
   sizeNext~size + size2, sizeNext~size + size2 + size3), makePlot = TRUE,
   legendPos = "bottomright", mainTitle = "Growth")
abline(a = 0, b = 1, lty= 2, col = "gray", lwd=2)
```

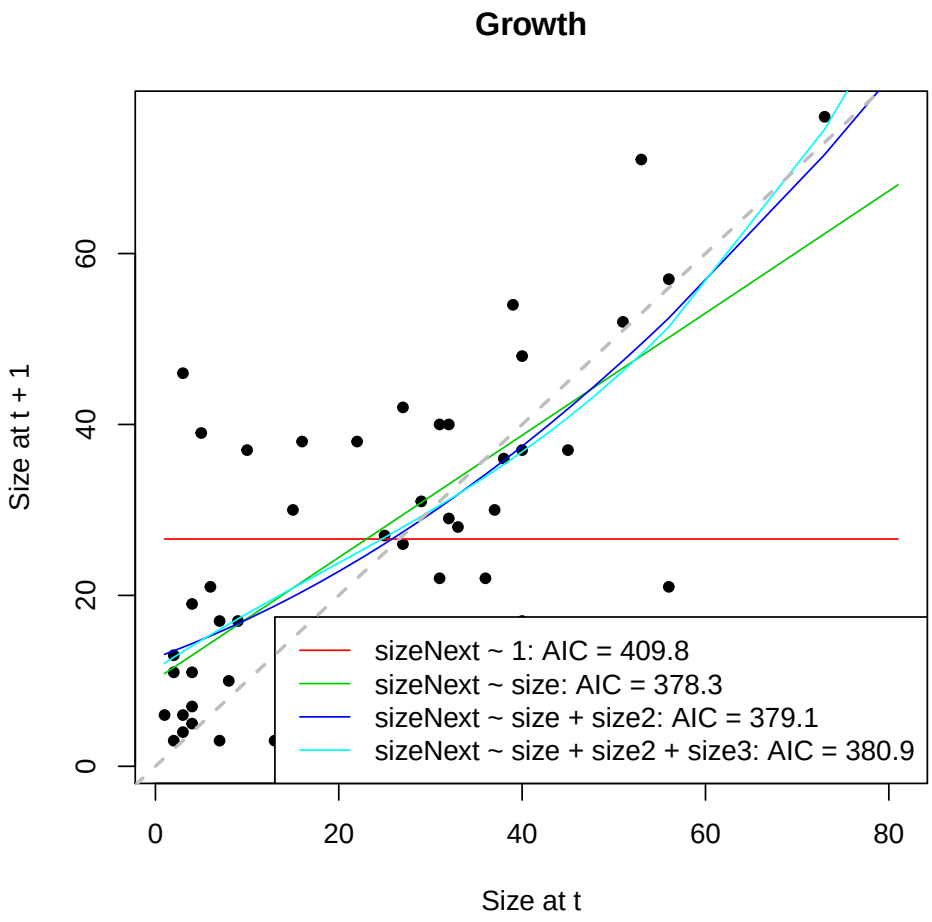

Figure 3.2: Change in size (maximum stem height) of *Hypericum cumulicola* individuals from 1994-95.

AIC values and knowledge of the species' life history suggest that a linear relationship between sizes in times t and $t+1$ adequately describes the process of growth.

```
go94 <- makeGrowthObj(d94, sizeNext~size)
```

With survival and growth objects in hand, the next step is to build a $\boldsymbol{P}$ matrix, which combines the survival and growth models into a single kernel. In this example, there is no evidence of eviction (See Section III.B. in the main text and Appendices *A, B*). Nonetheless, it is a good practice to choose *correction="constant"*, which prevents 'eviction' below *minSize* and above *maxSize* (see Williams et al. 2012).

```
Pmatrix94 <- makeIPMPmatrix(survObj = so94, growObj = go94,
```

```
                                minSize = minSize, maxSize = maxSize,
                                nBigMatrix = 80, correction = "constant")

require(fields)
image.plot(Pmatrix94@meshpoints,
           Pmatrix94@meshpoints,
           t(Pmatrix94),
           main = "Pmatrix: survival and growth",
           xlab = "Size at t",
           ylab = "Size at t+1")
   abline(a = 0, b = 1, lty= 2, col = "white", lwd=2)
```

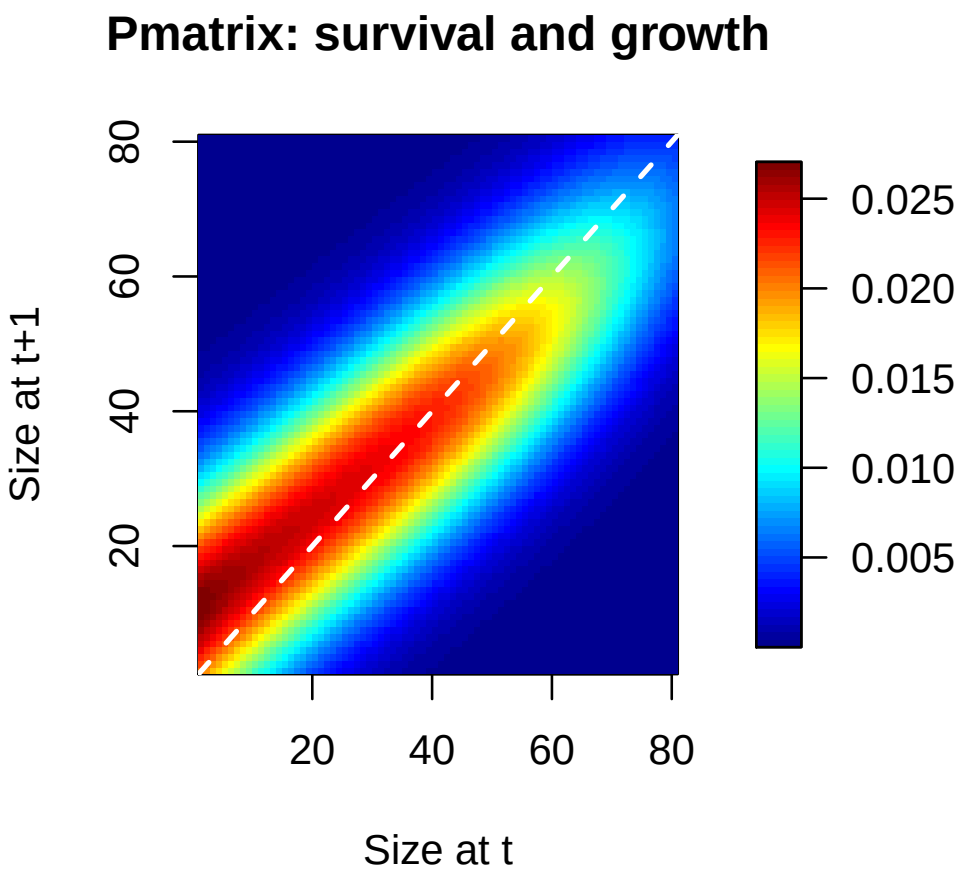

Figure 3.3: Survival and growth kernel, $\boldsymbol{P}$, of *Hypericum cumulicola* in 1994-1995.

A nice way to examine the adequacy of the range and breaks chosen for the continuous stage is to use the diagnostics available in *IPMpack*. Calling the function *PMatrixDiagnostics* yields a figure illustrating whether survival, life expectancy, and populations structure change with an increase in the number of bins or an increase in the size range. These diagnostics, among other aspects

discussed in Appendix *A* (*e.g.*, life expectancy), compare the stage-specific survival derived from the $\boldsymbol{P}$ matrix to the stage-specific survival in the survival object. Eviction at either end of the size range would result in lower estimated survival in the $\boldsymbol{P}$ matrix compared to the survival object. The close match between the red, black, and blue lines indicates that our model, as currently specified, is robust to accidental individual eviction.

```
diagnosticsPmatrix(Pmatrix94, growObj = go94, survObj = so94,
   correction = "constant")
[1] "Range of Pmatrix is "
[1] 1.529548e-08 2.705019e-02
[1] "Please hit any key for the next plot"
```

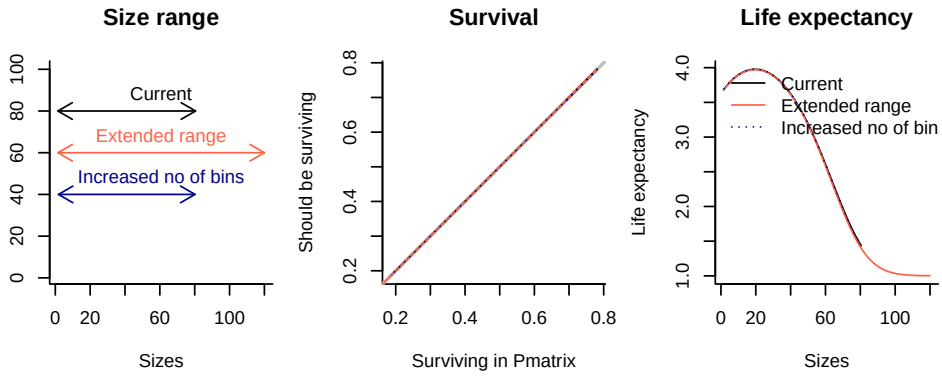

Figure 3.4: $\boldsymbol{P}$ matrix diagnostics.

3.4.2 Fecundity kernel

The last, and slightly more complicated step for building an IPM kernel is the creation of a fecundity ($\boldsymbol{F}$) kernel. Model-building and model comparison proceed as in the previous section for the $\boldsymbol{P}$ kernel. In this example, there are two vital rates that vary as a function of the continuous stage "maximum stem height": the probability of reproduction and the number of fruits per plant, labeled *fec0* and *fec1* respectively. Flowering probability (*fec0*) is modeled as a binomial response because it has only two possible outcomes: reproduction, or not. Fruit number (*fec1*) is modeled assuming a Poisson distribution because it derives from count (not continuous) data. *IPMpack 2.0* does not have a fecundity model comparison function (this function will be available in future versions), and so we will carry out model comparison manually:

```
fec0.0_94 <- makeFecObj(d94, Formula = fec0~1, Family = "binomial")
fec0.1_94 <- makeFecObj(d94, Formula = fec0~size, Family = "binomial")
fec0.2_94 <- makeFecObj(d94, Formula = fec0~size+size2, Family = "binomial")
fec0.3_94 <- makeFecObj(d94, Formula = fec0~size+size2+size3, Family = "binomial")
```

We then plot these models (Fig. 3.5).

```
pfzTest <- tapply(d94$size, as.numeric(cut(d94$size, 21)), mean, na.rm = TRUE)
psTest <- tapply(d94$fec0, as.numeric(cut(d94$fec0, 21)), mean, na.rm = TRUE)
fs <- order(d94$size); fs.fec0 <- (d94$fec0)[fs]; fs.size <- (d94$size)[fs]
pfz <- tapply(fs.size, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
ps <- tapply(fs.fec0, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
plot(as.numeric(pfz), as.numeric(ps), pch = 19, cex = 1, col = "black",
      ylim = c(0, 1), xlab = "size", ylab = "Proportion of reproductive individuals")
    y0 <- predict(fec0.0_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0)/(exp(y0)+1); lines(x, y0, col = 2)
    y0 <- predict(fec0.1_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0)/(exp(y0)+1); lines(x, y0, col = 3)
    y0 <- predict(fec0.2_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0)/(exp(y0)+1); lines(x, y0, col = 4)
    y0 <- predict(fec0.3_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0)/(exp(y0)+1); lines(x, y0, col = 5)
legend("bottomright", legend = sprintf("%s: %s = %.1f",
         c("1","size","size+size2","size+size2+size3"), c("AIC"),
         c(AIC(fec0.0_94@fitFec[[1]]), AIC(fec0.1_94@fitFec[[1]]),
         AIC(fec0.2_94@fitFec[[1]]), AIC(fec0.3_94@fitFec[[1]]))),
        col = c(2:5), lty = 1, xjust = 1, bg = "white")
```

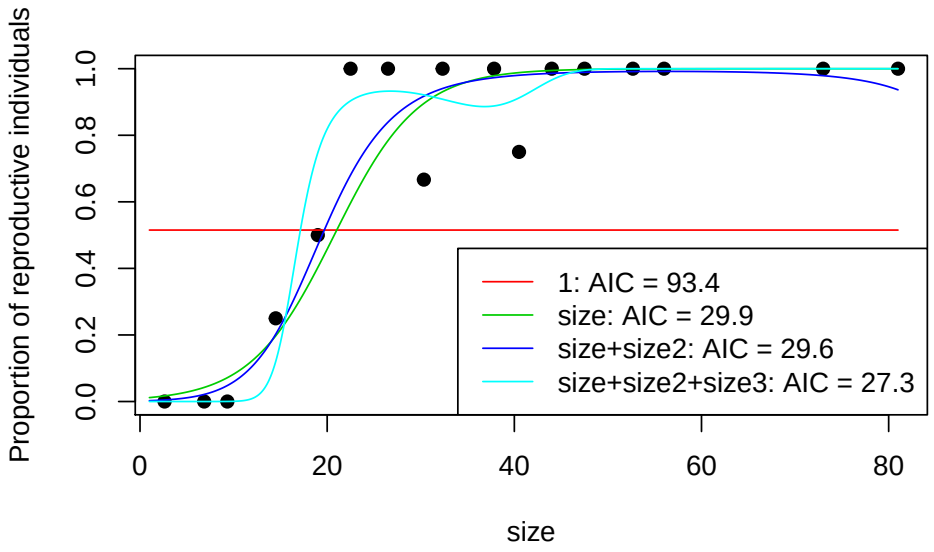

Figure 3.5: Model selection for flowering probability as a function of size for *Hypericum cumulicola* in 1994, and responses predicted by a series of regression models.

The cubic function of the vital rate *fec0* best fits the data, according to the AIC criterion, so we choose that model. In our experience, cubic functions work reasonably well with large herbs and sub-shrubs (see also the data example in *IPMpack* called *dataIPMpackCryptantha* for another case where survival and probability of reproduction data of a sub-shrub are best fit by a cubic function). A cubic function allows for a decrease of a probability in a binomial response (live vs. dead, reproduction vs. no reproduction) at intermediate values of the continuous stage, accommodating costly developmental changes that may occur, such as lignification of the stem, or first reproduction (Salguero-Gomez 2011).

```
fec1.0_94 <- makeFecObj(d94, Formula = fec1~1, Family = "poisson")
fec1.1_94 <- makeFecObj(d94, Formula = fec1~size, Family = "poisson")
fec1.2_94 <- makeFecObj(d94, Formula = fec1~size+size2, Family = "poisson")
fec1.3_94 <- makeFecObj(d94, Formula = fec1~size+size2+size3, Family = "poisson")
```

We examine an analogous series of regression models of *fec1*, the number of fruits per plant.

Visual and quantitative (*i.e.*, AIC scores) analyses of alternative models for the vital rate *fec1* indicate that a cubic function best fits the data, among the models considered.

We reemphasize that fecundity is almost always the most complicated part of an IPM; this is even more true when there is a discrete stage involved, like a seed bank. We encourage users to explore the help manual of the function *makeFecObj* to get a sense of the many options available for sexual reproduction within *IPMpack*.

```
fo94 <- makeFecObj(d94, Formula=c(fec0~size+size2, fec1~size+size2+size3),
                        Family=c("binomial", "poisson"),
                        Transform=c("none", -1),
                        meanOffspringSize=mean(d94[is.na(d1$size)==TRUE &
                            is.na(d94$sizeNext)==FALSE,"sizeNext"]),
                        sdOffspringSize=sd(d94[is.na(d1$size)==TRUE &
                            is.na(d94$sizeNext)==FALSE,"sizeNext"]),
                        fecConstants=data.frame(fec2=fec2,fec3=fec3,fec4=fec4),
                        offspringSplitter=data.frame(seedbank=goSB,
                          continuous=(1-goSB)),
                        vitalRatesPerOffspringType=data.frame(
                                                    seedbank=c(1,1,1,0,0),
                                                    continuous=c(1,1,1,1,1),
                                                    row.names=c("fec0","fec1",
                                                      "fec2","fec3","fec4")))
```

The *Formula* part within *makeFecObj* specifies the function, here, a cubic, describing the relationship between the vital rates (probability of reproduction, *fec0*, and number of fruits per plant, *fec1*) and the continuous stage (size). The next argument specifies the distribution assumed for the response (binomial, and Poisson, respectively). The following argument refers to whether the data require transformation or not. Because the vital rate *fec1* is conditional on individuals being

```
fs.fec1 <- (d94$fec1)[fs]
fs.size <- (d94$size)[fs]
pfz <- tapply(fs.size, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
ps <- tapply(fs.fec1, as.numeric(cut(fs.size, 21)), mean, na.rm = TRUE)
plot(as.numeric(pfz), as.numeric(ps), pch = 19, cex = 1, col = "black",
      xlab = "size", ylab = "Number of fruits/individual")
    y0 <- predict(fec1.0_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0); lines(x, y0, col = 2)
    y0 <- predict(fec1.1_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0); lines(x, y0, col = 3)
    y0 <- predict(fec1.2_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0); lines(x, y0, col = 4)
    y0 <- predict(fec1.3_94@fitFec[[1]], newdata = x0)
        y0 <- exp(y0); lines(x, y0, col = 5)
legend("topleft", legend = sprintf("%s: %s = %.1f",
       c("1","size","size+size2","size+size2+size3"), c("AIC"),
       c(AIC(fec1.0_94@fitFec[[1]]), AIC(fec1.1_94@fitFec[[1]]),
         AIC(fec1.2_94@fitFec[[1]]), AIC(fec1.3_94@fitFec[[1]]))),
        col = c(2:5), lty = 1, xjust = 1, bg = "white")
```

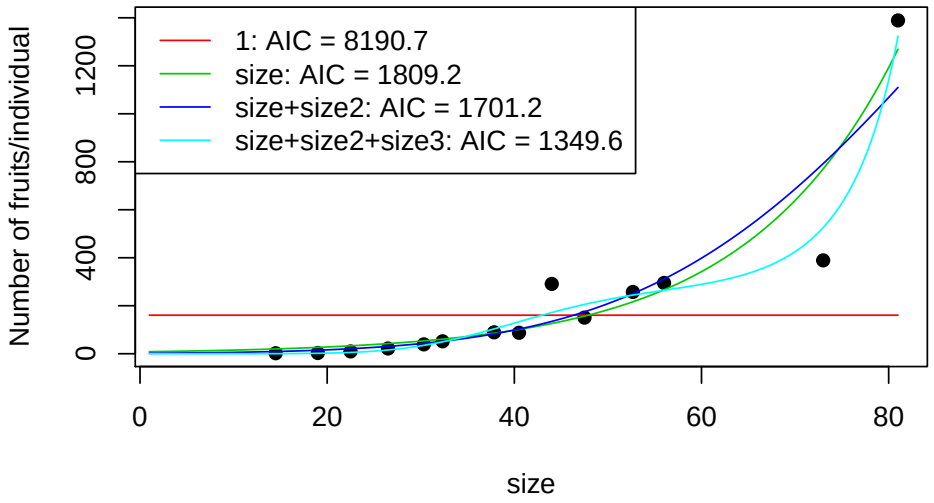

Figure 3.6: Model selection for the number of fruits per plant for *Hypericum cumulicola* in 1994, and responses predicted by a series of regression models.

reproductive (*i.e.*, *fec0 = 1*), and only contains values >1, we subtract a '1' as described in the example of *Succissa praetensis* in appendix *B*, i.e., i.e. fitting a translated Poisson model.

IPMpack automatically calculates the mean size and dispersion (*i.e.*, standard deviation) for new recruits into the continuous stage, referred to as *fec5* in the equation of $\boldsymbol{F}$ at the end of the first section of the present appendix. Note also that it is possible to assign other values for model parameters, to test or explore specific research questions. The argument *fecConstants* will store those vital rates that do not depend on the state variable (*i.e.*, size) - this includes *fec2, fec3* and *fec4*). Last, and perhaps the most interesting slots in the function *makeFecObj* for our modeling exercise with *Hypericum cumulicola*, are the following two arguments: *offspringSplitter* and *vitalRatesPerOffspringType*. The argument *offspringSplitter* determines the proportion of new propagules (here, seeds) being produced by individuals in the continuous stage that go to the discrete stage vs. the continuous stage. That is, seeds go into the seed bank with probability *goSB*) vs. germinate and become seedlings with probability (*1-goSB*). The final argument details which vital rates of fecundity (*fec 0 through 4*) affect propagules that go into the discrete stage, and which ones affect propagules recruited into the continuous stage, conditioned on their state (*i.e.*, size). Closer examination of the data frame will help make it evident that *fec0, fec1* and *fec3* affect all propagules, whereas *fec3* and *fec4* only affect seeds in the seed bank.

```
vitalRatesPerOffspringType=data.frame(seedbank=c(1,1,1,0,0),
                                      continuous=c(1,1,1,1,1),
                                      row.names=c("fec0","fec1","fec2","fec3","fec4"))
vitalRatesPerOffspringType
     seedbank continuous
fec0        1          1
fec1        1          1
fec2        1          1
fec3        0          1
fec4        0          1
```

The next step is to create the F kernel based on the fecundity object:

```
Fmatrix94 <- makeIPMFmatrix(fecObj=fo94, minSize=minSize, maxSize=maxSize,
                            nBigMatrix=80, correction="constant")
```

We plot the F kernel for the continuous stage.

```
require(fields)
image.plot(Fmatrix94@meshpoints,
           Fmatrix94@meshpoints,
           t(Fmatrix94[2:maxSize,2:maxSize]),
           main = "Fmatrix: fecundity",
           xlab = "Size at t",
           ylab = "Size at t+1")
```

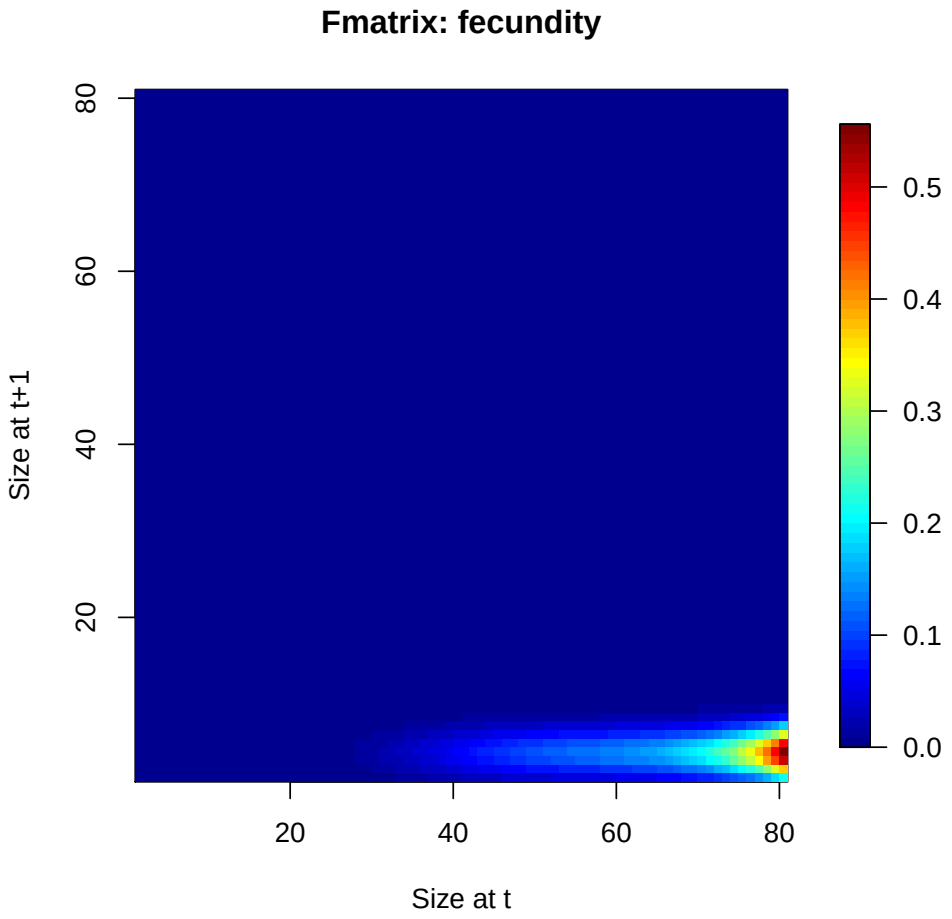

Figure 3.7: Fecundity kernel $\boldsymbol{F}$ of *Hypericum cumulicola* in 1994-1995 without the seed bank stage.

Now that we have constructed a $\boldsymbol{P}$ kernel for survival and growth with respect to size, and an $\boldsymbol{F}$ kernel for the contributions of reproductive individuals to both new seedlings and seeds in the seed bank, we must consider the transitions that take place into, within, and out of the discrete stage, the seed bank. To that end we will use the function *makeDiscreteTrans*.

```
dto94 <- makeDiscreteTrans(d94)
dummy94 <- as.matrix(fo94@offspringRel$coefficients[1])
dimnames(dummy94) <- list(1,"seedbank")
dto94@meanToCont <- as.matrix(dummy94,dimnames=c(1,"seedbank"))
```

```
dummy94 <- as.matrix(fo94@sdOffspringSize)
dimnames(dummy94) <- list(1,"seedbank")
dto94@sdToCont <- as.matrix(dummy94,dimnames=c(1,"seedbank"))
dto94@discreteTrans[1,1] <- staySB + (1 - staySB)*fec3*fec4
```

After having specified the transitions for the discrete stage, the seed bank, we can reconsider the P kernel, now including the discrete stage, with a single line:

```
Pmatrix94 <- makeIPMPmatrix(growObj=go94, survObj=so94, discreteTrans=dto94,
   minSize=minSize, maxSize=maxSize, nBigMatrix=80,correction="constant")
library(fields)
image.plot(c(Pmatrix94@meshpoints), c(Pmatrix94@meshpoints),
     (t(Pmatrix94[1:80,1:80])), main = "Pmatrix: survival/growth",
     xlab = "Size at t", ylab = "Size at t+1")
   abline(a = 0, b = 1, lty= 2, col = "white", lwd=2)
```

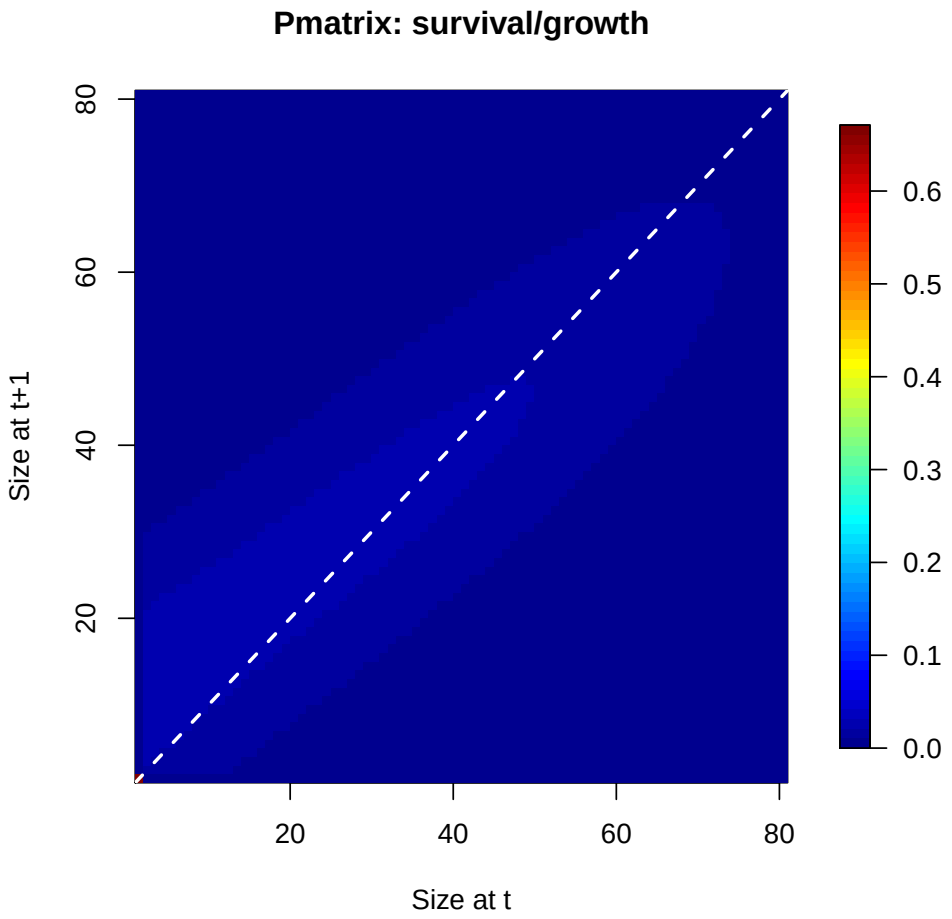

Figure 3.8: $\boldsymbol{P}$ kernel of *Hypericum cumulicola* in 1994-1995 with the seed bank stage. The row and column associated with the seed bank are the bottom-most and left-most, respectively. Although difficult to see, the bottom left-most cell, which corresponds to the probability of a seed remaining in the seed bank, has a value

Note that in Figure 3.8, the scale bar has a much higher scale than in the previous representation of the $\boldsymbol{P}$ kernel without seed bank. This is so due to the high probability of remaining in the seed bank in comparison to all other values in the kernel. The user might want to choose to represent this kernel on a log-scale. The same type of suggestion is rather useful when visually examining the $\boldsymbol{K}$ kernel, which includes the per-capita sexual contributions of the $\boldsymbol{F}$ kernel, and the transition probabilities of the $\boldsymbol{P}$ kernel.

At this point, creating an IPM simply consists of adding the $\boldsymbol{P}$ kernel and the $\boldsymbol{F}$ kernels together. Note that it is essential that the size range (i.e. *minSize*and *maxSize*) as well as the binning of the mesh (*nBigMatrix*) are identical in both subkernels.

```
IPM94 <- Pmatrix94 + Fmatrix94
```

Now that we have constructed an IPM kernel for the 1994-95 transition, we will construct an IPM kernel for the other pair of years of data, so that we can compare the two transitions and examine what drives the difference in population growth rates, using a Life Table Response Experiment (LTRE) analyses (Caswell 2001).

We define the data for the 1996-1997 transition as a subset of the original data object d1. Note again that the probabilities of transitioning between continuous and discrete stages are year-independent.

```
d96 <- subset(d1, d1$year == "1996" | d1$year == "All")
so96 <- makeSurvObj(d96, surv~size + size2)
go96 <- makeGrowthObj(d96, sizeNext~size)
fo96 <- makeFecObj(d96, Formula=c(fec0~size, fec1~size+size2),
                     Family=c("binomial", "poisson"),
                     Transform=c("none", -1),
                     meanOffspringSize=mean(d94[is.na(d1$size)==TRUE &
                         is.na(d94$sizeNext)==FALSE,"sizeNext"]),
                     sdOffspringSize=sd(d94[is.na(d1$size)==TRUE &
                         is.na(d94$sizeNext)==FALSE,"sizeNext"]),
                     fecConstants=data.frame(fec2=fec2,fec3=fec3,fec4=fec4),
                     offspringSplitter=data.frame(seedbank=goSB,
                       continuous=(1-goSB)),
                     vitalRatesPerOffspringType=data.frame(
                                              seedbank=c(1,1,1,0,0),
                                              continuous=c(1,1,1,1,1),
                                              row.names=c("fec0","fec1",
                                                "fec2","fec3","fec4")))
Fmatrix96 <- makeIPMFmatrix(fecObj = fo96, minSize = minSize, maxSize = maxSize,
                            nBigMatrix = 80, correction = "constant")
dto96 <- makeDiscreteTrans(d96)
dummy96 <- as.matrix(fo96@offspringRel$coefficients[1])
dimnames(dummy96) <- list(1,"seedbank")
dto96@meanToCont <- as.matrix(dummy96,dimnames=c(1,"seedbank"))
dummy96<-as.matrix(fo96@sdOffspringSize)
dimnames(dummy96) <- list(1,"seedbank")
dto96@sdToCont <- as.matrix(dummy96,dimnames=c(1,"seedbank"))
dto96@discreteTrans[1,1] <- staySB + (1 - staySB)*fec3*fec4
Pmatrix96 <- makeIPMPmatrix(growObj = go96, survObj = so96, discreteTrans = dto96,
                            minSize = minSize, maxSize = maxSize, nBigMatrix = 80,
                            correction="constant")
IPM96 <- Pmatrix96 + Fmatrix96
```

3.4.3 Life Table Response Experiment analyses

The comparison of population dynamics under different conditions is one of the most valuable approaches to understand ecological and evolutionary processes. Life Table Response Experiment (LTRE) analyses are designed for such data; they are a widely-used method of decomposition of variation in stage-structured demography (Caswell 2001) and have been applied to IPMs (Williams and Crone 2006; Williams et al. 2010; Shou-Li et al. 2011; Gonzalez et al. 2012; Bassar et al. 2013; Yule et al. 2013). LTREs require at least two different stage-structured population models. We begin by examining basic output from the two IPMs created for *Hypericum cumulicola*. The population growth rate (λ) is important because it gives an indication of how individuals in a population are performing on average (McGraw & Caswell 1996). For the two IPMs that we created from data on *H. cumulicola*, the corresponding population growth rates are similar:

```
 lambda94 <- Re(eigen(IPM94)$value[1])
 lambda96 <- Re(eigen(IPM96)$value[1])
 lambda94
[1] 0.8922246
 lambda96
[1] 0.7660452
```

The significance of the difference between these growth rates could be tested by creating bootstrapped replicates of the *d94* vs. *d96* subsets of the data, hundreds of times, examining the distribution of λ for each, and calculating their overlap. We shall not demonstrate this here because it is not the primary goal of this appendix, though. Instead, we will assume that the two population growth rates are significantly different.

Why then is the population growth rate from 1996-1997 lower than that from the 1994-1995? There are several possible explanations: perhaps individuals in 1996-1997 had higher mortality; perhaps they grew less, whereby their reproductive output was lower too (let us remember that in this species the probability of reproduction and number of fruits produced increase with size as shown above); or perhaps the recruits produced were smaller and were associated with a higher probability of mortality. Some combination of the above is possible. The two IPMs are shown in the figure below.

```
require(fields)
par(mfrow=c(1,2))
image.plot(2:dim(IPM94)[1],
           2:dim(IPM94)[1],
           t(IPM94[2:maxSize,2:maxSize]),
           main = "IPM 1994-1995",
           xlab = "Size at t",
           ylab = "Size at t+1")
abline(a = 0, b = 1, lty= 2, col = "white", lwd=2)
image.plot(2:dim(IPM96)[1],
           2:dim(IPM96)[1],
           t(IPM96[2:maxSize,2:maxSize]),
           main = "IPM 1996-1997",
           xlab = "Size at t",
           ylab = "Size at t+1")
abline(a = 0, b = 1, lty= 2, col = "white", lwd=2)
```

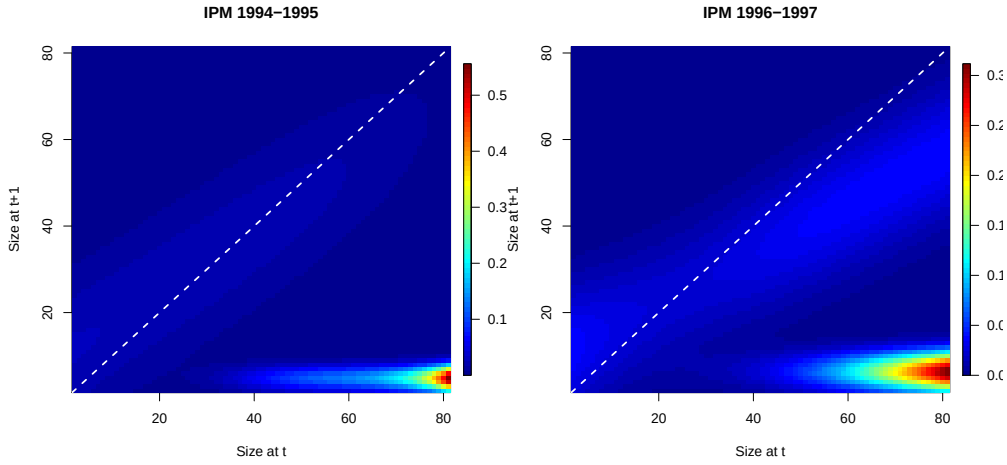

Figure 3.9: (left) IPM $\boldsymbol{K}$ kernels for 1994-1995 (right) and 1996-1997 (left) for *Hypericum cumulicola*. Note the difference in color-coding scale between the two panels.

In the following, we perform a one-way LTRE analysis (for more sophisticated designs, read Caswell 2001, Caswell 2010, and Davison et al. 2010). The basic idea behind an LTRE is to construct a linear model where the difference between estimated population growth rates arises from the additive effects of the differences in the underlying vital rates, weighted by their associated sensitivities (Caswell et al. 1984, de Kroon et al. 1986, 2000).

The first step is to define the baseline (control) for our comparisons. For convenience, we will arbitrarily define the IPM for 1994-1995 as the 'control'; thus we are interested in examining how the differences in λ from the IPM in 1996-1997 in comparison to the IPM 'control' are brought about by the underlying processes of survival, growth and reproduction. In what follows we adhere to the terminology used by Caswell (2001, p. 260) for one-way fixed designs in LTREs. The only difference is that instead of using the term 'matrix' ($\boldsymbol{A}$), we use the term 'IPM' (or kernel $\boldsymbol{K}$).

The second step in the LTRE is to designate the arithmetic mean of the two IPMs as IPM $K^{\dagger}$. This mid-way IPM will be used later on to evaluate the differences in the sensitivities of the IPM kernels to be compared.

$$K^{\dagger} = \frac{(K^{(treatment)} + K^{(control)})}{2} \tag{3.3}$$

```
IPM_mid <- (IPM94 + IPM96)/2
```

The population growth rate λ of the treatment IPM, which in our case is the 1996-1997 transition, can be defined using the following approximation, where a_{ij} refers to cell (i,j) of the IPM matrix:

$$\lambda_{1996-1997} = \lambda_{1996-1997} + \sum_{i,j}(a_{ij}^{(m)} - a_{ij}^{(r)}) \frac{\delta\lambda}{\delta(a_{ij})}\bigg|_{K^{\dagger}} \tag{3.4}$$

The next step in the LTRE is to calculate the IPM of the differences between the $\boldsymbol{K}$ kernels corresponding to the 1994-1995 and 1996-1997 transitions:

$$K_{Differences} = K_{1996-1997} - K_{1994-1995} \tag{3.5}$$

```
IPM_diff <- IPM96 - IPM94
```

Next, we inspect the parts of the life cycle of the study species that are responsible for the difference in λ between 1994-1995 and 1996-1997, by weighting the IPM of the difference between those two IPMs by the sensitivity of the arithmetic mean IPM:

$$K_{Contributions} = K_{Differences} \times S_{K_{\dagger}} \tag{3.6}$$

```
Sensi_IPM_mid <- sens(IPM_mid)
IPM_contrib <- IPM_diff * Sensi_IPM_mid
```

To interpret the results, we will use the *image* function of the library *fields*, as we did to plot the kernels $\boldsymbol{P}$, $\boldsymbol{F}$ and $\boldsymbol{K}$:

```
require(fields)
par(mfrow=c(1,3))
image.plot(0:dim(Sensi_IPM_mid)[1],
           0:dim(Sensi_IPM_mid)[1],
           t(log(Sensi_IPM_mid)),
           main = "Sensitivity of mid-way IPM",
           xlab = "Seedbank and size at t",
           ylab = "Seedbank and size at t+1")
abline(a = 0, b = 1, lty= 2, col = "black", lwd=2)
legend("topleft",legend="A",box.lwd = 0)
image.plot(2:dim(IPM_diff)[1],
           2:dim(IPM_diff)[1],
           t(IPM_diff[2:maxSize,2:maxSize]),
           main = "Element differences between both IPMs",
           xlab = "Size at t",
           ylab = "Size at t+1")
abline(a = 0, b = 1, lty= 2, col = "black", lwd=2)
legend("topleft",legend="B",box.lwd = 0)
image.plot(2:dim(IPM_contrib)[1],
           2:dim(IPM_contrib)[1],
           t(IPM_contrib[2:maxSize,2:maxSize]),
           main = "Contributions to differences\nin expression(lambda)",
           xlab = "Size at t",
           ylab = "Size at t+1")
abline(a = 0, b = 1, lty= 2, col = "black", lwd=2)
legend("topleft",legend="C",box.lwd = 0)
```

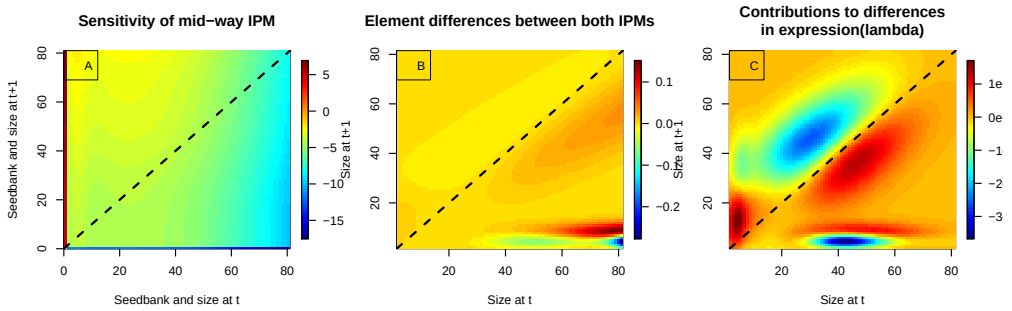

Figure 3.10: (A) Sensitivity of the arithmetic mean IPM $K^{\dagger}$ kernel, including seed bank dynamics; (B) difference between the kernels $K_{1994-1995}$ and $K_{1996-1997}$; (C) contributions to the difference in λ between the two *Hypericum cumulicola* IPMs.

Figure 3.10.A shows the importance of the emergence of seeds from the seed bank to λ of the arithmetic mean IPM $K^{\dagger}$ kernel. That is, the sensitivity of the IPM $K^{\dagger}$ kernel to changes in the emergence of seeds from the seed bank is high (see de Kroon 1986, 2000 for a more detailed interpretation of sensitivities and elasticities). In contrast, the graduation of new seeds into the seed bank has very low absolute impact on λ. The importance of seeds emerging from the seed bank is not because seeds can become individuals of 20-80 cm height in a single year; this does not

happen. Moreover, we assumed that the population dynamics in the seed bank are the same in both 1994-1995 and 1996-1997 K kernels (*dto94* and *dto96*). The seed bank dynamics will cancel out in the kernel of differences, and thus their contribution to the difference between $\lambda_{1996-1997}$ and $\lambda_{1994-1995}$ will be null. With these constraints in mind, Figures 9.B and 9.C display kernels for just the continuous stage. Figure 9.C illustrates that two processes underlie the fact that $\lambda_{1996-1997}$ is lower than $\lambda_{1994-1995}$. First, there was more shrinkage of individuals of 20-60 cm height in 1996-1997 in comparison to the population in 1994-1995. Let us remember that in this species the probability of reproduction (*fec0*) and the number of seeds produced per plant (*fec1*) increase with size, so a decrease in size results in fewer seeds produced by time *t+1* (see also Figure 10 below). Second, individuals of large sizes produced more and larger seedlings in the 1996-1997 interval than individuals in 1994-1997. However, because the sensitivity of this latter process is negligible (figure 9, left), the difference in shrinkage is the leading cause of the difference in λ.

A recommended cross-check to verify that the LTRE was done correctly is that the summation of the contributions of the matrix elements in the discretized kernel of $IPM_{Contributions}$ should approximately equal the differences between the λ values for $IPM_{1996-1997}$ and $IPM_{1994-1995}$.

$$\lambda_{1996-1997} = \lambda_{1994-1995} + \sum_{i,j} c_{ij} \tag{3.7}$$

```
 lambda96 - lambda94
[1] -0.1261795
 sum(IPM_contrib)
[1] -0.1233634
```

To our knowledge, no tests or statistics have been designed to evaluate the adequacy of the approximation used in LTREs. Instead, deviations from the above equation of less than 5% are taken as 'good enough' in a rule-of-thumb manner. Here, the deviation is only 1.45%.

```
 lambda_diff <- lambda96 - lambda94
 LTRE_sum_contrib <- sum(IPM_contrib)
 abs(lambda_diff-LTRE_sum_contrib)/abs(lambda_diff)*100
[1] 2.231843
```

The user can also perform LTREs at the level of the whole vital rates (e.g. survival, fec0, fec1, etc), or even at the underlying parameters substituting the command *sens* with *sensParams*.

3.5 Conclusions

IPMs are becoming a popular tool for modeling the dynamics of structured populations (>80 plant and animal IPM publications in the last decade; R. Salguero-Gomez, unpublished data). Researchers value the fact that no artificial boundaries have to be imposed onto the life cycle of a species to study its demography (Easterling et al. 2000). Nonetheless, IPMs can indeed accommodate the need to consider the role of discrete stages in the life cycle. Many species have a life history that is a combination of continuous stages (e.g. height, DBH, weight) and discrete stages

(*e.g.* seed bank, instars, etc). Alternatively, the choice to include one or more discrete stages may result from an inability to measure a continuous variable that is a good proxy for survival, changes within that stage or among stages, and reproduction. Regardless of the reason for considering discrete stages in the life history of a given study species, IPMs as implemented in *IPMpack* allow us to parameterize life histories using both continuous and discrete stages, as exemplified here using a perennial plant with a seed bank.

3.6 Acknowledgements

The demographic data for *Hypericum cumulicola* used in this appendix were collected by Pedro Quintana-Ascencio (UCF) and Eric Menges (Archbold Biological Station).

3.7 References

- Bassar, R.D., Lopez-Sepulcre, A., Reznick, D.N. & Travis, J. (2013). Experimental Evidence for Density-Dependent Regulation and Selection on Trinidadian Guppy Life Histories. The American Naturalist, 181, 25âĂŞ38.
- Caswell, H. (2001). Matrix population models: construction, analysis, and interpretation. Sinauer, Sunderland, MA.
- Caswell, H. (2000). Prospective and retrospective perturbation analyses: their roles in conservation biology. Ecology, 81, 619-627.
- Caswell, H., Brault, S., Read, J.J., & Smith, T.D. (1984). Evaluating the consequences of reproduction in complex salmonid life cycles. Aquaculture, 43, 123-134.
- Easterling, M. R., Ellner, S. P., & Dixon, P. M. (2000). Size-specific sensitivity: applying a new structured population model. Ecology, 81, 694-708.
- Eager, E. A., Haridas, C. V., Pilson, D., Rebarber, R., & Tenhumberg, B. (2013). Disturbance frequency and vertical distribution of seeds affect long-term population dynamics: a mechanistic seed bank model. The American Naturalist, 182, 180-190.
- de Kroon, H., van Groenendael, J.M. & Ehrlen, J. (2000). Elasticities: a review of methods and model limitations. Ecology, 81, 607-618.
- de Kroon, H., Plaisier, A., van Groenendael, J. & Caswell, H. (1986). Elasticity: the relative contribution of demographic parameters to population growth rate. Ecology, 67, 1427âĂŞ1431.
- Gonzalez, E.J., Rees, M. & Martorell, C. (2012). Identifying the demographic processes relevant for species conservation in human-impacted areas: does the model matter? Oecologia, 171, 347âĂŞ356.

- Jacquemyn, H., Brys, R. & Jongejans, E. (2010). Size-dependent flowering and costs of reproduction affect population dynamics in a tuberous perennial woodland orchid. Journal of Ecology, 98, 1204-1215.
- Metcalf, C., McMahon, S.M., Salguero-Gomez, R. & Jongejans, E. (2013). IPMpack: an R package for integral projection models. Methods in Ecology and Evolution, 4, 195-200.
- Quintana-Ascencio, P.A., Menges, E.S. & Weekley, C.W. (2003). A fire-explicit population viability analysis of *Hypericum cumulicola* in Florida rosemary scrub. Conservation Biology, 17, 433-449.
- Salguero-Gomez, R. (2011) Physiological bases of plant shrinkage and its demographic implications. PhD thesis. UPenn Press, USA.
- Li, S.-L., Yu, F.-H., Werger, M.J., Dong, M. & Zuidema, P.A. (2011). Habitat-specific demography across dune fixation stages in a semi-arid sandland: understanding the expansion, stabilization and decline of a dominant shrub. Journal of Ecology, 99, 610-620.
- Williams, J. & Crone, E. (2006). The impact of invasive grasses on the population growth of Anemone patens, a long-lived native forb. Ecology, 87, 3200-3208.
- Williams, J., Auge, H. & Maron, J. (2010). Testing hypotheses for exotic plant success: parallel experiments in the native and introduced ranges. Ecology, 91, 1355-1366.
- Williams, J.L., Miller, T.E., & Ellner, S.P. (2012) Avoiding unintentional eviction from integral projection models. Ecology, 93, 2008-2014.
- Yule, K.M., Miller, T.E.X. & Rudgers, J.A. (2013). Costs, benefits, and loss of vertically transmitted symbionts affect host population dynamics. Oikos, in press.

Chapter 4

Appendix D: Vital rate regressions for IPMs

contact: Sean M. McMahon (mcmahons@si.edu)

Abstract

This appendix gives an overview of how regression models of vital rates are built and combined into population projections using IPMs. It focuses on models of trees, which present some important challenges to building IPMs and highlight the ways in which vital rate models are (and are not) critical to projections. Fundamentally, IPMs with trees can be sensitive to vital rate models because trees can live a long time, grow very slowly, and die rarely when old. After introducing the concept of regression models as the core of IPMs, we build IPMs for trees and test sensitivities of population statistics from IPMs (life expectancy and passage time) to survival and growth model structure and sample sizes. Overall, inference can be sensitive to growth model form, but expected growth given size is less important to IPMs than the variance structure in growth. Survival model structure is more sensitive to the sampling of large individuals, where stochastic mortality patterns can have a strong influence on model form. As sample size increases, models will tend to capture biologically meaningful mortality patterns, but determining that sampling size is not possible. We recommend testing sensitivities to sample size by sub-sampling from data and reproducing multiple IPMs. This demonstrates trends towards stability. Our sampling results have implications for all models of population projection of trees (individual based models, simulators, etc.) as survival models estimated from data may be highly sensitive to sampling. .

4.1 Introduction

Most of the strengths of IPMs are strengths of regressions. Each vital rate is regressed against individual states and covariates. Appropriate models include linear and Generalized Linear Models, (GLMs) Generalized Additive Models (GAMs) for arbitrarily smooth relationships, and hierarchical or multilevel models for clustered data with varying sample sizes (Bolker et al. 2013). Consequently, models can be constructed quickly using standard variable selection methods and model diagnostics, using established software and procedures. The simplicity of regression facilitates an iterative approach to modeling, allowing researchers to move between data, vital rate models, and population-level predictions in order to arrive at robust and reliable IPMs.

4.2 Vital Rate Models

Modeling vital rates often poses a challenge for large-scale demographic studies that generalize across populations, among species, or along environmental gradients, because data are missing for some values of the state variable or the covariates. When collecting more data is not feasible, it is necessary to infer vital rates for non-sampled states or environmental contexts. Such inference is naturally performed with regression. For example, by building a logistic regression for survival probability as a function of size, one can infer the survival probability of an individual of any size, not simply those observed. The quality of population inference thus lies in the biologically plausibility of the vital rate functions for inferred transitions.

Below, we discuss size dependence of vital rates using different classes of regression for illustration and use the general terms β_0 and β_1 to refer to regression intercepts and slopes, respectively.

4.2.1 State transitions: Linear Models and distributional assumptions

Linear models assuming normality in the residual error term are typically used to characterize size transitions, i.e. growth, $g(z',z)$, and the recruit size distribution, $f_{recruit}(z',z)$. For example, parameterizing the growth function, $g(z',z)$, requires modeling the distribution of possible sizes at the next census, z', where at least one of the predictors involve the current size, z. The simplest approach describes the expected size of individuals, $\bar{z}'$, as $\bar{z}' = \beta_0 + \beta_1 z$. The predicted variation around this (conditional) mean is captured by the residual variance from the fitted model, σ^2. Thus the growth function can be written as a normal distribution:

$$g(z',z) = Ce^{-\frac{(z'-\bar{z}')^2}{2\sigma^2}} \tag{4.1}$$

where C is the normalizing constant for the normal distribution. This growth model assumes that the residual variance is independent of expected size. If the variance depends on size, one can transform size (e.g., log or square-root) or use a generalized least squares approach to model the relationship between the expected mean size and the variance. Figure 1b,c shows a model where σ^2 is a linear function of z (and $\bar{z}'$ is also a function of z^2). The distributional assumptions

(e.g. constant variance) are particularly important when modelling state transitions, because they directly influence the form of the resulting IPM. In contrast, distributional assumptions are less critical when modelling the probabilities and rates discussed below, because the parameters of the 'error term' do not appear in the IPM.

The same type of model underpins the recruit size distribution, $f_{recruit}(z', z)$, though it is often simpler because it only contains an intercept term in $\bar{z}'$ if offspring size is assumed to be independent of maternal size. Often, $f_{recruit}(z')$ is modeled as a normal or lognormal distribution with mean and variance equal to observed mean and variance of recruits at $t+1$ (e.g., Ellner and Rees 2006). When pedigree can be inferred or vegetative reproduction is observed, offspring size can be modeled as a function of maternal size (Easterling et al. 2000, Coulson et al. 2011, Appendix H).

4.2.2 Probabilities and rates: demography and the generalized linear model

Many vital rates describe a probability (e.g., survival) or a count (e.g. the number of offspring), which are naturally modeled as a function of traits using generalized linear models. For example, survival data can be modeled as a Bernoulli random variable (1 or 0, denoting survival or death, respectively), where survival probability depends on individual size using logistic regression: $logit(p_{surv}) = \beta_0 + \beta_1 z$. Similar models can be constructed for other life history transitions, with a multinomial model replacing the binomial model when there are more than two categories. For example, one could model the probability of transitioning between discrete stages (e.g., metamorphosis in arthropods) or between continuous stages (e.g., between seedlings, whose sizes are often measured as height, and adult trees, whose sizes are often measured as diameter). Other vital rates that enter the individual component of the kernel, such as germination or recruitment probability, can be modeled using only an intercept if size dependence can be ignored.

Similarly, count data describing reproductive output (e.g., number of seeds) is often modeled using a Poisson regression, which takes the form $f_{seeds}(z) = e^{\beta_0 + \beta_1 z}$. If the data are over dispersed âĂŞ the variance in seed production is greater than the mean âĂŞ one can use a negative binomial distribution. If the expected number of offspring is large (> 100), it may be reasonable to transform the data and use a linear model (Dahlgren et al. 2011). Proximate measures of recruitment can be used if the total offspring number is unavailable. For instance, one can model the number of flowering rosettes per individual and multiply this by the average number of seeds per rosette (Salguero-Gomez et al. 2012).

4.2.3 Nonlinear relationships

When modeling the relationship between a vital rate and size, slight departures from linearity can have important effects on population-level predictions (Dahlgren and Ehrlen 2009; Ozgul et al. 2010). This is not surprising, as many physiological and metabolic relationships are nonlinear, and vital rate changes against a variable like size should be, in many organisms and systems, nonlinear. Modeling nonlinear relationships between states and vital rates can be addressed in a variety of ways. Transforming the state variable is sometimes sufficient. If explicitly modeling

nonlinear responses is necessary, a straightforward approach is using generalized linear models (GLMs) which can compare a series of nested polynomial models of increasing degree to capture the nonlinearity. Unfortunately, the predictive performance of polynomial regressions outside the range of data is often poor, though these problems can be improved using fractional polynomial regression (Sauerbrei et al. 2007). More sophisticated nonlinear methods include non-parametric local regression (e.g. restricted cubic splines; Dahlgren et al. 2009) and semi-parametric methods such as GAMs (Hastie and Tibshirani 1986; Ozgul et al. 2010). The flexibility of GAMs are particularly well suited for IPMs, as they can accommodate the same range of 'error' distributions as GLMs, mixtures of parametric and nonparametric components in the linear predictor and can automatically choose the degree of smoothing.

4.2.4 Borrowing strength

When data are clustered into clear groups (such as populations at different sites), hierarchical models offer a tool for simultaneously estimating regression parameters for all subgroups (including groups with few data). For a detailed discussion of this modeling approach in general, we recommend Gelman and Hill (2007). In effect, a hierarchical analysis, estimates parameters for regressions for each group. These parameters, however, are themselves estimated from a distribution constructed across all data (a 'pooled' estimate), where parameters for groups with large sample sizes are weighted more from their own data, while parameter estimates for groups with lower samples are weighted by the overall pooled estimate. In the context of IPMs, this can estimate parameters for regressions of vital rates on traits that offer the Best Linear Unbiased Predictor (BLUP) that can give sensible results, even for sparse data. This approach can account for unmeasured differences between subgroups, such as populations, years, locations, or species (Ellner and Rees 2006; Dahlgren et al. 2009; Nicole et al. 2011). In practice, hierarchical models can eliminate marginally significant predictors that would be otherwise retained by a nonhierarchical model, leading to simpler, more generalizable models (Latimer et al. 2006).

Hierarchical models can, as mentioned above, improve parameter estimates across populations with unbalanced samples, which is particularly important if some life stages are rare but disproportionally important to the overall population dynamics. For example, the survival of large canopy trees is critical to seed production, but for rare species there are few individuals providing information on an inherently stochastic process. Often, mortality is not observed among large individuals, implying immortality at large size classes. Hierarchical models can adjust this biased expectation by incorporating estimates from other groups with sufficient sample sizes to appropriately estimate mortality rates of large individuals. This can help create biologically realistic estimates of survival for rare species. It is important to note that this will not help determine differences among species, because rare species will, by statistical necessity, have similar parameter sets as the most common species. As long as this shortcoming is understood, hierarchical models offer an approach that might be preferable to the alternatives: removing rare species from analyses altogether or allowing the data to dictate unrealistic life-history patterns.

4.2.5 Predictors

A myriad of predictor variables can be added to the basic vital rate functions described above to better explain variation in vital rates. Additional state variables might include age (Childs et al. 2003; Ellner and Rees 2006), sex, infection status (Bruno et al. 2010), genotype (Coulson et al. 2011), individual quality (Ellner and Rees 2006), and even space, to describe spatially explicit dispersal (Jongejans et al. 2011). Covariates further predict the relationship between state variables and vital rates, e.g., how size-dependent growth varies during succession (Metcalf et al. 2009). Covariates can also include trait differences (e.g. specific leaf area), abiotic environments (Section IV; Dalgren and Ehrlen 2009; Dalgleish et al. 2011, Nicole et al. 2011), time lags in the effect of covariates on vital rates (Kuss et al. 2008), and competition (Adler et al. 2010), to mention a few.

4.3 Study system/objectives

Demographic data for trees are most commonly collected in permanent sample plots. The data come in the form of censuses, where in each censusevery tree above a certain diameter is tagged and measured (or found and re-measured), and all stems are mapped. The interval between censuses can vary from 1 year to over a decade. These data offer a way to model tree population change based on the 'snap-shots' of censuses. However, due to the limited temporal extent of these data (even long-term plots will provide data on only a portion of a canopy tree's life), some care must be taken when building models of vital rates. The best statistical models may 'fit' the data, but provide misleading inference when those vital rate models are incorporated in projections. Here we explore an example of tree demography using data from permanent sample plots in vital rate models, and then incorporate those models into IPMs.

4.4 Data

Census data were obtained from a public archive (http://esapubs.org/archive/ecol/ E092/115/default.htm#data, Pelissier et al. 2011). The total data set consists of repeated measurements across 102 tree species over 20 years (1989-2010) in the Uppangala Permanent Sample Plot of old-growth wet evergreen Dipterocarp forest in the Western Ghats of India (details are in the metadata for PÃl'lissier et al. 2011). Data reported include the dates of measurement, bole girth measured at breast height (GBH) and whether individuals survived or not between census intervals. The census intervals varied around a mean of four years (Fig. 1). For this example we use a single species, Myristica dactyloides (Myristicaceae). We converted girth measurements to diameter at breast height (DBH) for ease of comparison with other studies. Further we corrected the growth increment (the size difference between successive samples) for measurements at different time periods to standardize it at four year intervals.

4.5 Analysis

4.5.1 Sensitivity of survival parameters to model structure and sample size

As mentioned above, survival regression models can be sensitive to patterns of tree death, especially with higher-order terms or species with few trees in some size classes (where rare mortality events can skew parameter estimates). Here we show how sample sizes influence model fits for survival models with and without a polynomial term. As survival is a binary variable, we can linearize the regression with the inverse logit function for easier notation. The first model run is:

$$logit^{-1}P(s) = \alpha + \beta size, \tag{4.2}$$

where the response is the inverse logit function on the probability of survival in a time interval ($P(s)$), α is the intercept and *size* is the only covariate and β is the coefficient that relates the influence of size on the probability of survival. The second model includes a $size^2$ term. To simultaneously test the influence of sample size on these models, we run regressions with twenty samples of a subset of the data (approximately 75%, 50%, and 25% of the data (actual N is listed in the panel titles). Figures 1 and 2 show the sample size fits for the two regressions (without and then with $size^2$, respectively).

```
library(IPMpack)
dff1 <- read.csv(file = "Tree_data.csv", header = TRUE)
dff <- dff1[complete.cases(dff1$size, dff1$sizeNext, dff1$incr), ] #(omit NAs)
dff <- subset(dff, incr > 0) # remove negative growth
ncuts <- 20 # number of bins for visualizing the survival data
n.samp <- 50 # number of samples drawn for the sample size analysis
# Size only in regresion
dff.ln <- dim(dff1)[1]
samp.size <- c(dff.ln, round(dff.ln * 0.75), round(dff.ln * 0.5), round(dff.ln *
 0.25))
par(mfrow = c(2,2))
for(ss in 1:length(samp.size)) {
   samp.obs <- dff1[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
   sv1 <- makeSurvObj(samp.obs, Formula = surv ~ size)
   if(ss == 1) {
     picSurv(samp.obs, sv1, 20, makeTitle = "Complete Data")
   }else{
     os <- order(dff1$size)
     os.surv <- (dff1$surv)[os]
     os.size <- (dff1$size)[os]
     psz <- tapply(os.size, as.numeric(cut(os.size, ncuts)), mean, na.rm = TRUE)
     ps <- tapply(os.surv, as.numeric(cut(os.size, ncuts)), mean, na.rm = TRUE)
```

```
    plot(as.numeric(psz), as.numeric(ps), pch = 19, xlab = "Size at t",
      ylab ="Survival to t+1",main = sprintf("%i Observations", samp.size[ss]))

    for(i in 1:n.samp) {
      samp.obs <- dff1[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
      sv1 <- makeSurvObj(samp.obs, Formula = surv ~ size)
      lines(samp.obs$size[order(samp.obs$size)],
      surv(samp.obs$size[order(samp.obs$size)], data.frame(covariate = 1),
        sv1), type= "l", col = "steelblue")
    }
    points(as.numeric(psz), as.numeric(ps), pch = 19, col = "tomato")
  }
}
```

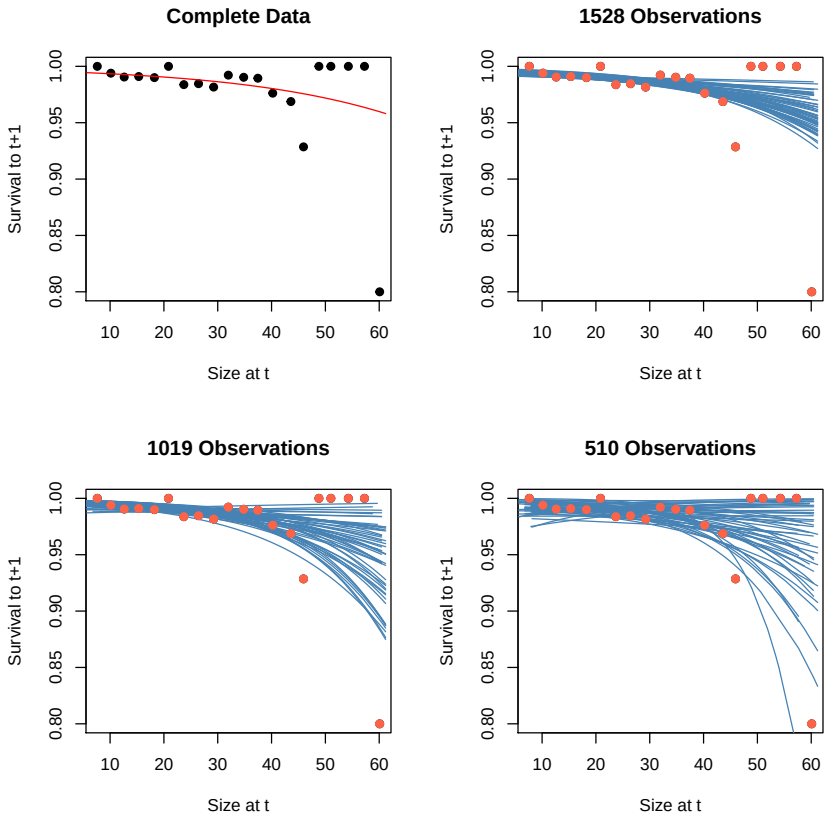

Figure 4.1: Sampling survival models with increment regressed against size. Fits with a) all of the data and b-d) subsamples of the data re-sampled 20 times.

```
##  Size + Size ^ 2
dff1 <- read.csv(file = "Tree_data.csv", header = TRUE)
dff <- dff1[complete.cases(dff1$size, dff1$sizeNext, dff1$incr), ]# omit NAs
dff <- subset(dff, incr > 0) # remove negative growth
dff.ln <- dim(dff1)[1]
samp.size <- c(dff.ln, round(dff.ln * 0.75), round(dff.ln * 0.5), round(dff.ln *
 0.25))
par(mfrow = c(2,2))
for(ss in 1:length(samp.size)){
   samp.obs <- dff1[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
   sv1 <- makeSurvObj(samp.obs, Formula = surv ~ size + size2)
   if(ss == 1) {
     picSurv(samp.obs, sv1, 20, mainTitle = "Complete Data")
   }else{
     os <- order(dff1$size)
     os.surv <- (dff1$surv)[os]
     os.size <- (dff1$size)[os]
     psz <- tapply(os.size, as.numeric(cut(os.size, ncuts)), mean, na.rm = TRUE)
     ps <- tapply(os.surv, as.numeric(cut(os.size, ncuts)), mean, na.rm = TRUE)
     plot(as.numeric(psz), as.numeric(ps), pch = 19, xlab = "Size at t",ylab
       ="Survival to t+1",main = sprintf("%i Observations", (samp.size[ss])))

     for(i in 1:n.samp) {
       samp.obs <- dff1[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
       sv1 <- makeSurvObj(samp.obs, Formula = surv ~ size + size2)
       lines(samp.obs$size[order(samp.obs$size)],
       surv(samp.obs$size[order(samp.obs$size)], data.frame(covariate = 1),
         sv1), type= "l", col = "steelblue")
     }
     points(as.numeric(psz), as.numeric(ps), pch = 19, col = "tomato")
   }
 }
```

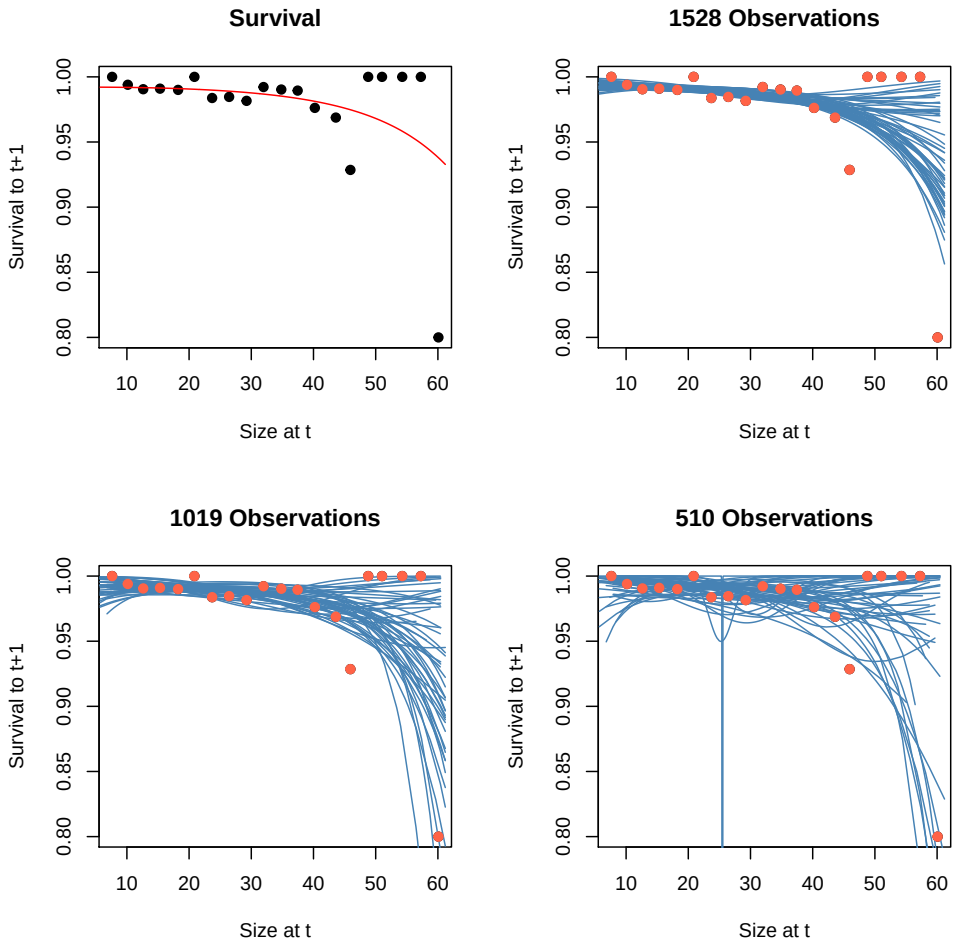

Figure 4.2: Sampling survival models with increment regressed against size and size^2. Fits with a) all of the data and b-c) subsamples of the data re-sampled 20 times.

4.5.2 Sensitivity of growth parameters to model structure and sample size

We repeated the exercise for growth models with regression structures the same as in 4.2, except that the left side of the equation is growth increment. Figures 4.3 and 4.4 show the plotted fits and demonstrate that they are less sensitive to sample size than survival, where the form is conserved even at low samples. It is worth noting that the functional form when using a polynomial term allows the generally positive growth trend with size to reverse at large sizes. This senescence can be important to population models (tested below), and also may reflect mechanisms important to population dynamics (e.g., vulnerability of large trees to disturbance). The risk, of course, is that the flexibility afforded by that term, may with low sample sizes lead to an increase in growth at the highest sizes (which can be seen in some of the blue lines in Fig. 4.4). As with the survival model, doing this bootstrap with the data (even if using the full dataset, but bootstrapping it with replacement) can indicate whether the actual collected samples show sensitivity to being sampled.

```
dff1 <- read.csv(file = "Tree_data.csv", header = TRUE)
#dff will be for growth (no NAs)
dff <- dff1[complete.cases(dff1$size, dff1$sizeNext, dff1$incr), ]
dff <- subset(dff, incr > 0) # remove negative growth
# Size only in regresion
n.samp <- 50
dff.ln <- dim(dff)[1]
samp.size <- c(dff.ln, round(dff.ln * 0.75), round(dff.ln * 0.5), round(dff.ln *
 0.25))
par(mfrow = c(2,2))
for(ss in 1:length(samp.size)) {
   samp.obs <- dff[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
   gr1 <- makeGrowthObj(samp.obs, Formula = incr ~ size)
   if(ss == 1) {
     picGrow(samp.obs, gr1, mainTitle = "Complete Data")
   }else{
     plot(dff$size, dff$incr, col = "lightgray", pch = 19, xlab = "Size at t",
 ylab = "Growth increment",  main = sprintf("%i Observations", samp.size[ss]))
   }
     for(i in 1:n.samp) {
       samp.obs <- dff[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
       gr1 <- makeGrowthObj(samp.obs, Formula = incr ~ size)
       new.size <- seq(0, max(dff$size), length = 100)
       pred.grow <- predict(gr1@fit, newdata = data.frame(size = new.size))
       points(new.size, pred.grow, type = "l", col = "steelblue")
     }
 }
```

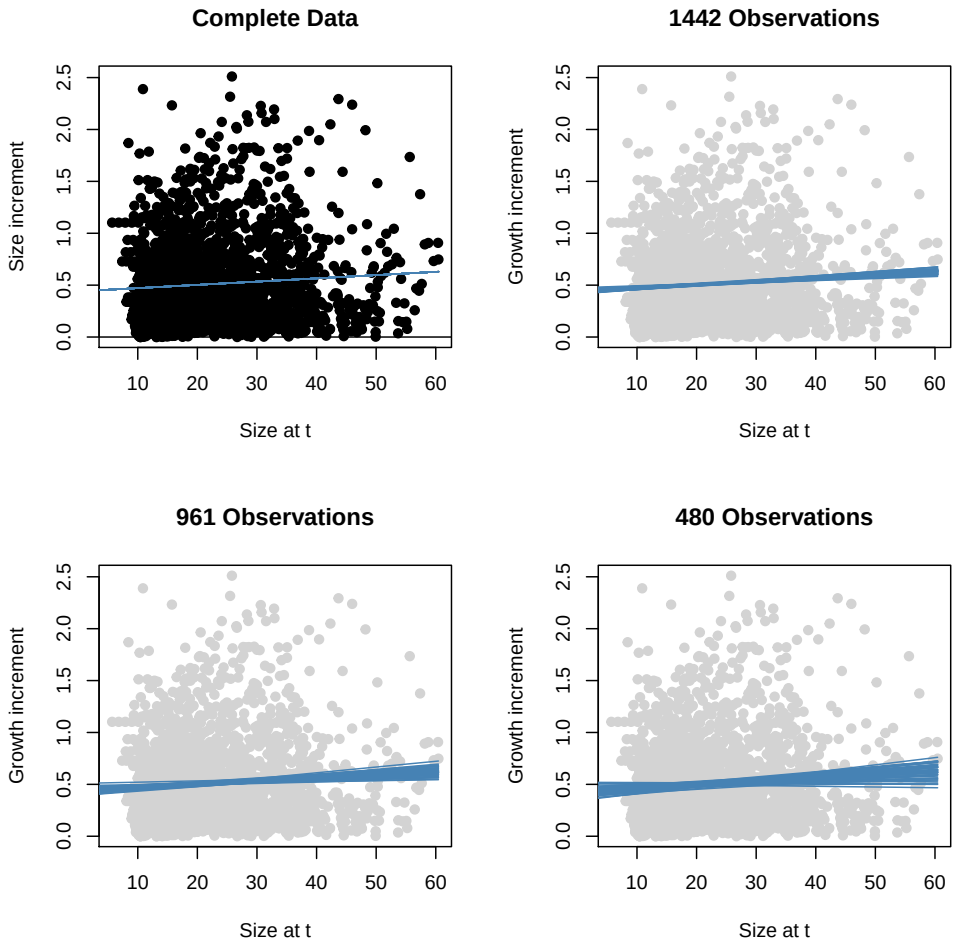

Figure 4.3: Sampling growth models with increment regressed against size. Fits with a) all of the data and b-c) subsamples of the data re-sampled 20 times.

```
dff1 <- read.csv(file = "Tree_data.csv", header = TRUE)
#dff will be for growth (no NAs)
dff <- dff1[complete.cases(dff1$size, dff1$sizeNext, dff1$incr), ]
dff <- subset(dff, incr > 0) # remove negative growth
# Size only in regresion
n.samp <- 50
dff.ln <- dim(dff)[1]
samp.size <- c(dff.ln, round(dff.ln * 0.75), round(dff.ln * 0.5),
   round(dff.ln * 0.25))
par(mfrow = c(2,2))
for(ss in 1:length(samp.size)) {
   samp.obs <- dff[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
   gr1 <- makeGrowthObj(samp.obs, Formula = incr ~ size + size2)
   if(ss == 1) {
     picGrow(samp.obs, gr1, mainTitle = "Complete Data")
   }else{
     plot(dff$size, dff$incr, col = "lightgray", pch = 19, xlab = "Size at t",
 ylab = "Growth increment",  main = sprintf("%i Observations", samp.size[ss]))

     for(i in 1:n.samp) {
       samp.obs <- dff[sample(seq(dff.ln), samp.size[ss], replace = FALSE) ,]
       gr1 <- makeGrowthObj(samp.obs, Formula = incr ~ size + size2)
       new.size <- seq(min(dff$size), max(dff$size), length = 100)
       pred.grow <- predict(gr1@fit, newdata = data.frame(size = new.size, size2
 = I(new.size)^2))
       points(new.size, pred.grow, type = "l", col = "steelblue")
     }
   }
 }
```

4.5.3 Integrating the sensitivities of vital rate models

IPMs combine vital rate models of growth and survival into a P Matrix. This matrix essentially combines the sensitivities of the models to covariates and sample sizes shown above. Even without a fecundity vital rate model (F Matrix), we can build IPMs and use them to project important population-level metrics, such as longevity and passage time. In this section we repeat the efforts of the last to sections but continue forward down the IPM workflow and actually create IPMs and derive estimates of passage time and longevity for models with different covariate structures and

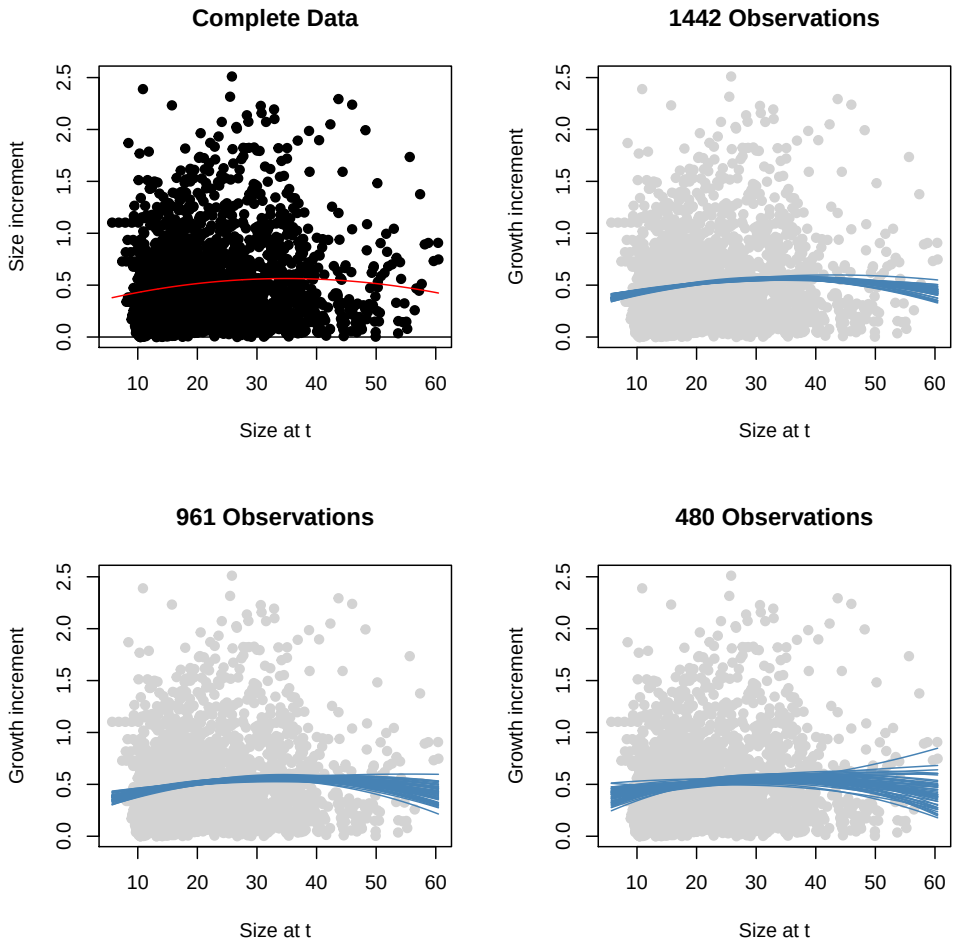

Figure 4.4: Sampling growth models with increment regressed against size and size^2. Fits with a) all of the data and b-c) subsamples of the data re-sampled 20 times.

sample sizes. Because growth shows little sensitivity to sample size we use growth models with the squared term only.

```
# Size only in regresion
dff1 <- read.csv(file = "Tree_data.csv", header = TRUE)
#dff will include only trees that grew (no NAs or shrinkers)
dff <- dff1[complete.cases(dff1$size, dff1$sizeNext, dff1$incr), ]
dff <- subset(dff, incr > 0) # remove negative growth
n.samp <- 50
dff.ln.g <- dim(dff)[1]
dff.ln.s <- dim(dff1)[1]
samp.size.g <- c(dff.ln.g, round(dff.ln.g * 0.75), round(dff.ln.g * 0.5),
 round(dff.ln.g * 0.25))
samp.size.s <- c(dff.ln.s, round(dff.ln.s * 0.75), round(dff.ln.s * 0.5),
```

```
 round(dff.ln.s * 0.25))
matSize <- 150
gr1 <- makeGrowthObj(dff, Formula = incr ~ size + size2)
sv1 <- makeSurvObj(dff1, Formula = surv ~ size)
sv2 <- makeSurvObj(dff1, Formula = surv ~ size + size2)
IPM.1 <- makeIPMPmatrix(nBigMatrix = matSize, minSize = 7, maxSize =
 max(dff1$sizeNext, na.rm = TRUE), growObj = gr1, survObj = sv1, correction =
 "constant", integrateType = "cumul")
IPM.2 <- makeIPMPmatrix(nBigMatrix = matSize, minSize = 7, maxSize =
 max(dff1$sizeNext, na.rm = TRUE), growObj = gr1, survObj = sv2, correction =
 "constant", integrateType = "cumul")
samp.list <- vector("list", length = 3)
for(ss in 2:length(samp.size.g)) {

   IPM.1.pt <- matrix(NA, n.samp, matSize)
   IPM.2.pt <- matrix(NA, n.samp, matSize)
   IPM.1.le <- matrix(NA, n.samp, matSize)
   IPM.2.le <- matrix(NA, n.samp, matSize)

   for(i in 1:n.samp) {
     samp.obs.g <- dff[sample(seq(dff.ln.g), samp.size.g[ss], replace = FALSE) ,]
     samp.obs.s <- dff1[sample(seq(dff.ln.s), samp.size.s[ss],
                               replace = FALSE), ]
     gr1 <- makeGrowthObj(samp.obs.g, Formula = incr ~ size + size2)
     sv1 <- makeSurvObj(samp.obs.s, Formula = surv ~ size)
     sv2 <- makeSurvObj(samp.obs.s, Formula = surv ~ size + size2)
     IPM.1.tmp <- makeIPMPmatrix(nBigMatrix = matSize, minSize = 7, maxSize =
 max(dff1$sizeNext, na.rm = TRUE), growObj = gr1, survObj = sv1, correction =
 "constant", integrateType = "cumul")
     IPM.2.tmp <- makeIPMPmatrix(nBigMatrix = matSize, minSize = 7, maxSize =
 max(dff1$sizeNext, na.rm = TRUE), growObj = gr1, survObj = sv2, correction =
 "constant", integrateType = "cumul")
     # note multiplication by 4 to account for census interval
     IPM.1.pt[i,] <- passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.1.tmp) * 4
     IPM.2.pt[i,] <- passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.2.tmp) * 4
     IPM.1.le[i,] <- meanLifeExpect(IPM.1.tmp) * 4
     IPM.2.le[i,] <- meanLifeExpect(IPM.2.tmp) * 4
   }
   OP.list <- list(IPM.1.pt = IPM.1.pt, IPM.2.pt = IPM.2.pt, IPM.1.le = IPM.1.le,
```

```
    IPM.2.le = IPM.2.le)
  samp.list[[ss - 1]] <- OP.list
 }
par(mfcol = c(2,3))
for(s in 1:3) {

  plot(IPM.1@meshpoints, passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.1) *
 4, type = "l", xlab = "Size (cm)", ylab = "Years", main = sprintf("Size: %i
 Observations", samp.size[s + 1]), ylim = c(0, 500))

  for(i in 1:n.samp) {
    lines(IPM.1@meshpoints, samp.list[[s]]$IPM.1.pt[i,], col = "steelblue")
  }
  lines(IPM.1@meshpoints, passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.1) *
 4, lwd = 1.5, col = "red")

  plot(IPM.2@meshpoints, passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.2) *
 4, type = "l", xlab = "Size (cm)", ylab = "Years", main = sprintf("Size +
 Size^2: %i Observations", samp.size[s + 1]), ylim = c(0, 500))

  for(i in 1:n.samp) {
    lines(IPM.2@meshpoints, samp.list[[s]]$IPM.2.pt[i,], col = "steelblue")
  }
  lines(IPM.1@meshpoints, passageTime(max(dff$sizeNext, na.rm = TRUE), IPM.2) *
 4, lwd = 1.5, col = "red")
 }
```

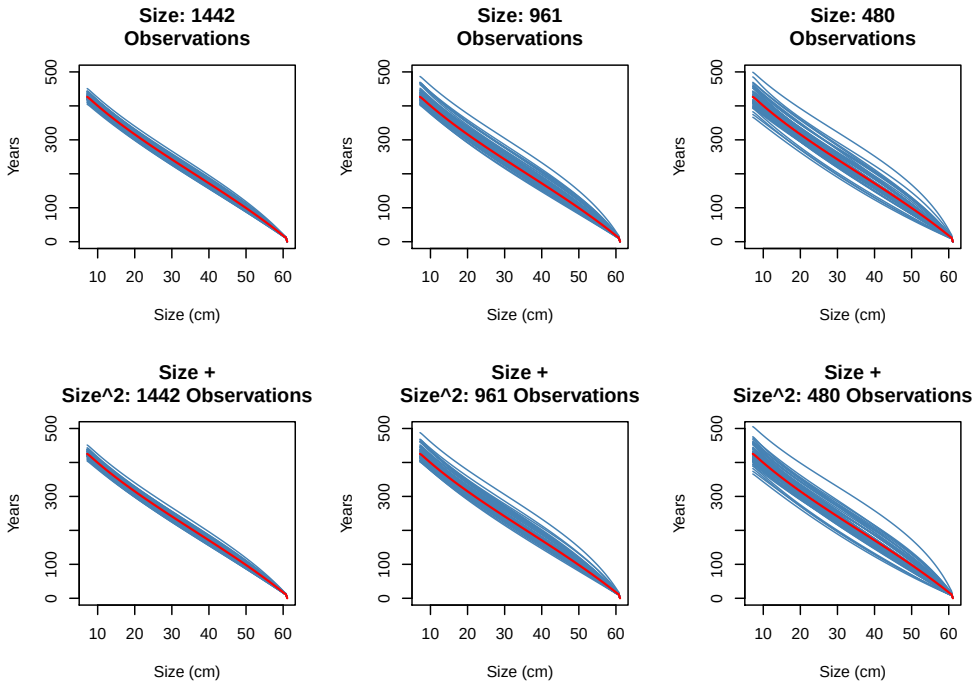

Figure 4.5: Passage time by model structure and sample size. The top row of panels show models with *size* as the only covariate, and the bottom row includes $size^2$. Columns show 50 fits with 75%, 50%, and 25% of the data. The red line shows the estimate with all of the data.

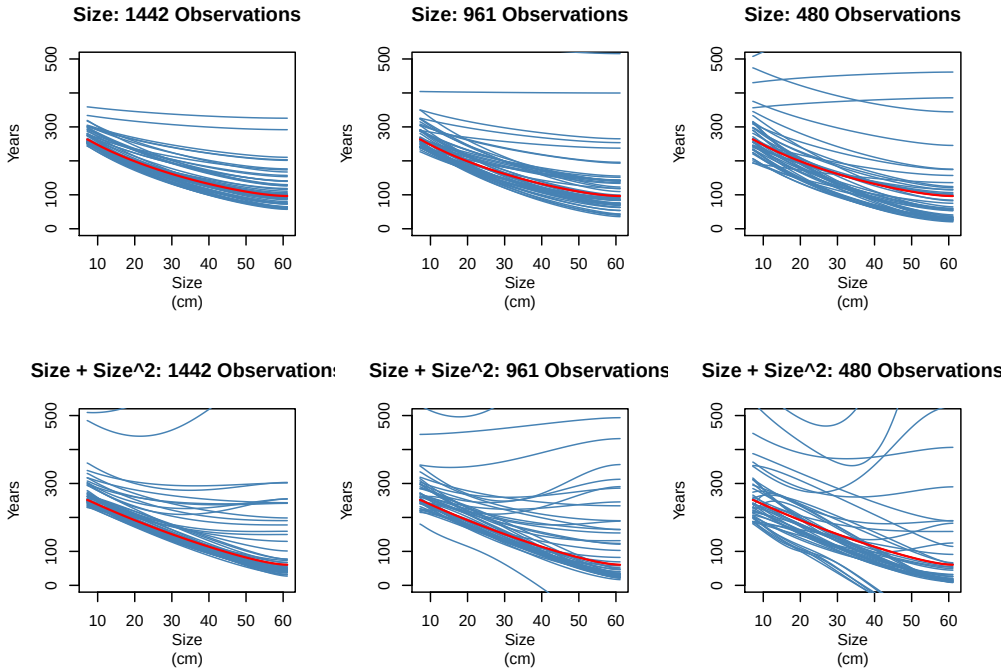

Figure 4.6: Life expectancy by model structure and sample size. The top row of panels show models with *size* as the only covariate, and the bottom row includes $size^2$. Columns show 50 fits with 75%, 50%, and 25% of the data. The red line shows the estimate with all of the data.

```
par(mfcol = c(2,3))
for(s in 1:3) {
   plot(IPM.1@meshpoints, meanLifeExpect(IPM.1) * 4, type = "l", xlab = "Size
 (cm)", ylab = "Years", main = sprintf("Size: %i Observations", samp.size[s +
 1]), ylim = c(0, 500))
   for(i in 1:n.samp) {
     lines(IPM.1@meshpoints, samp.list[[s]]$IPM.1.le[i,], col = "steelblue")
   }
   lines(IPM.1@meshpoints, meanLifeExpect(IPM.1) * 4, lwd = 1.5, col = "red")

   plot(IPM.2@meshpoints, meanLifeExpect(IPM.2) * 4, type = "l", xlab = "Size
 (cm)", ylab = "Years", main = sprintf("Size + Size^2: %i Observations",
 samp.size[s + 1]), ylim = c(0, 500))
   for(i in 1:n.samp) {
     lines(IPM.2@meshpoints, samp.list[[s]]$IPM.2.le[i,], col = "steelblue")
   }
   lines(IPM.2@meshpoints, meanLifeExpect(IPM.2) * 4, lwd = 1.5, col = "red")
 }
```

Figures 4.5 and 4.6 show how changing sample size and model structure influence population dynamics estimated from IPM output. For this slow-growing and high surviving species, passage time to maximum diameter begins around 400 years and declines monotonically. There appear little difference in model structure due to the higher-order term for this species, but one can see how the higher order term might change in model behavior as the when sample sizes are smaller, passage times estimated early in life can vary by over 150 years. The form remains fairly stable. Survival has a much more direct influence on estimates of mean life expectancy given size. Estimates begin around 270 years and declines monotonically. Unlike passage time, here the higher order term can make a huge difference in predicted dynamics when low sample sizes allow the stochastic death rate to create unrealistic survival curves. Putting priors on the higher order term (such as a community mean) might be wise if sample size analysis shows sensitivity through bootstraps.

4.6 Conclusions

Regressions form the backbone of IPMs. As such, all of the ways in which regressions can be built, tested, and projected work with IPMs. This flexibility allows the IPMs to be less of a 'black box' where demographic information goes in and population projections emerge. Instead, IPMs can be used to test assumptions about vital rate models, sample sizes, and sensitivities. In fact, building an IPM should involve moving back and forth between vital rate models and IPM inference so that at end the user has a robust understanding of the vital rates that drive populations as well as the population patterns that emerge from the IPMs.

4.7 References

- Adler, P.B., Ellner, S.P.& Levine, J.M. (2010). Coexistence of perennial plants: an embarrassment of niches. Ecology Letters, 13, 1019-1029.
- Bolker, B.M., et al (2013). Strategies for fitting nonlinear ecological models in R, AD Model Builder, and BUGS. Methods in Ecology and Evolution, in press.
- Bruno, J., Ellner, S., Vu, I., Kim, K.& Harvell, C. (2011). Impacts of aspergillosis on sea fan coral demography: modeling a moving target. Ecological Monographs, 81, 123-139.
- Childs, D.Z., Rees, M., Rose, K.E., Grubb, P.J.& Ellner, S.P. (2003). Evolution of complex flowering strategies: an age- and size-structured integral projection model. Proceedings of the Royal Society B: Biological Sciences, 270, 1829-1838.
- Coulson, T., MacNulty, D.R., Stahler, D.R., vonHoldt, B., Wayne, R.K.& Smith, D.W. (2011). Modeling effects of environmental change on wolf population dynamics, trait evolution, and life history. Science, 334, 1275-1278.
- Dahlgren, J.P.& Ehrlen, J. (2009). Linking environmental variation to population dynamics of a forest herb. Journal of Ecology, 97, 666-674.
- Dahlgren, J. (2011). Nonlinear relationships between vital rates and state variables in demographic models. Ecology, 92, 1-7.
- Easterling, M., Ellner, S.& Dixon, P. (2000). Size-specific sensitivity: applying a new structured population model. Ecology, 81, 694-708.
- Ellner, S.& Rees, M. (2006). Integral projection models for species with complex demography. American Naturalist, 167, 410-428.
- Gelman, A. & Hill, J. (2007). Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge Univ Press, Cambridge, UK.

- Jongejans, E., Sheppard, A.W.& Shea, K. (2006). What controls the population dynamics of the invasive thistle Carduus nutans in its native range? Journal of Applied Ecology, 43, 877-886.

- Latimer, A.M., Wu, S., Gelfand, A.E.& Silander, J.A., Jr. (2006). Building statistical models to analyze species distributions. Ecological applications : a publication of the Ecological Society of America, 16, 33-50.

- Kuss, P., Rees, M., AegisdÃşttir, H.H., Ellner, S.P.& Stocklin, J. (2008). Evolutionary demography of long-lived monocarpic perennials: a time-lagged integral projection model. Journal of Ecology, 96, 821-832.

- Metcalf, C., Horvitz, C., Tuljapurkar, S.& Clark, D. (2009b). A time to grow and a time to die: a new way to analyze the dynamics of size, light, age, and death of tropical trees. Ecology, 90, 2766-2778.

- Nicole, F., Dahlgren, J.P., Vivat, A., Till-Bottraud, I.& EhrlÃl'n, J. (2011). Interdependent effects of habitat quality and climate on population growth of an endangered plant. Journal of Ecology, 99, 1211-1218.

- Ozgul, A., Childs, D.Z., Oli, M.K., Armitage, K.B., Blumstein, D.T., Olson, L.E., Tuljapurkar, S.& Coulson, T. (2010). Coupled dynamics of body mass and population growth in response to environmental change. Nature, 466, 482-485.

- Pelissier, R. Pascal, J-P., Ayyappan, N., Ramesh, B. R., Aravajy, S. Ramalingam, S.R. (2011). Twenty years tree demography in an undisturbed Dipterocarp permanent sample plot at Uppangala, Western Ghats of India. Ecology 92:1376.

- Salguero-Gomez, R., Siewert, W., Casper, B.B.& Tielborger, K. (2012). A demographic approach to study effects of climate change in desert plants. Philosophical Transactions of the Royal Society B: Biological Sciences, 367, 3100-3114.

- Sauerbrei W., Meier-Hirmer C., Benner A. & Royston P. (2006). Multivariable regression model building by using fractional polynomials: description of SAS, STATA and R programs, Computational Statistics and Data Analysis, 50: 3464-3485.

Chapter 5

Appendix E: IPMs in varying environments

Contact: C. Jessica Metcalf (charlotte.metcalf@zoo.ox.ac.uk)

Abstract

No environment is stable. This usually affects vital rates, with consequences for population structure and growth. Here we introduce three tools for modeling population dynamics in fluctuating environments - the analysis of transient dynamics, calculation of the stochastic population growth rate, and mega-matrix modeling for environments changing in a Markovian fashion - using Integral Projection Models.

5.1 Introduction

Demographic rates may fluctuate because of changes in both the abiotic (droughts, freezes,etc.) and abiotic environments (herbivory, etc.), with consequences for population dynamics. This appendix briefly outlines some approaches for exploring these consequences using IPMs. We will start by providing a very simple exploration of transient dynamics, followed by an example of modeling stochastic dynamics. We end by illustrating the 'mega-matrix' approach for modeling a population experiencing a Markovian sequence of environments.

5.2 Study system/objectives

The aim is to i) consider (in the simplest context) transient dynamics; ii) show how to estimate the stochastic rate of increase; and iii) introduce concepts that could be used to explore quantities such as the stochastic life expectancy.

5.3 Data

We use data simulated to represent a long-lived perennial plant to examine transient dynamics and calculate of the long term stochastic growth rate of a population experiencing a stochastically-varying environment. The demographic consequences of a changing light environment is illustrated using real data on canopy trees at La Selva Biological Station in Cost Rica (data available at http://esapubs.org/archive/ecol/E093/019/metadata.htm).

5.4 Analyses

5.4.1 Transient dynamics

There are many sophisticated approaches for analyzing the transient dynamics of matrix population models (MPM's; see Caswell, 2007; Haridas &Tuljapurkar, 2007). These can be extended directly to IPMs (Vindenes et al. 2011; Ozgul et al. 2012; Yule et al. 2013). Here we demonstrate the simplest way of exploring transient dynamics, i.e., by simulating population dynamics using an IPM.

First, we simulate some data and build an IPM using the R package *IPMpack*:

```
library(IPMpack)
# create data
dff<-generateData()
# make growth, survival and fertility objects
gr1 <- makeGrowthObj(dataf=dff,Formula=sizeNext~size+size2)
sv1 <- makeSurvObj(dataf=dff, Formula=surv~size+size2)
fv1 <- makeFecObj(dff,Transform="log")
# build an IPM
Pmat <- makeIPMPmatrix(nBigMatrix=100,minSize=-3,maxSize=14,
          growObj=gr1,survObj=sv1, correction="constant")
Fmat <- makeIPMFmatrix(nBigMatrix=100,minSize=-3,maxSize=14,
          fecObj=fv1,correction="constant")
IPM <- Pmat
IPM@.Data <- Pmat+Fmat
```

Note that we build the IPM by over-writing the data slot in a Pmatrix with the sum of the Pmatrix and Fmatrix - this allows us to retain the extra information in the Pmatrix (the meshpoints identity, etc).

Now generate an initial population composed of 1000 individuals, with sizes randomly drawn from a normal distribution. These data are put into a format that can be multiplied by the IPM, i.e., the number of individuals in each of the IPM's size bins:

```
# generate a starting population
startingSizes <- rnorm(1000,mean=5,sd=3)
```

```
# identify the breakpoints of the IPM (midpoints -h/2 to max midpoint + h/2)
breakpoints <- c(IPM@meshpoints-(IPM@meshpoints[2]-IPM@meshpoints[1])/2,
                 IPM@meshpoints[length(IPM@meshpoints)]+
                 (IPM@meshpoints[2]-IPM@meshpoints[1])/2)
# put the startingSizes into discrete bins that follow those of the IPM from the breakpoints
index.new.dist <- findInterval(startingSizes,breakpoints,all.inside=TRUE)
loc.sizes <- table(index.new.dist);
n.new.dist0 <- rep(0,length(IPM@meshpoints))
n.new.dist0[as.numeric(names(loc.sizes))] <- loc.sizes
```

We now use matrix multiplication to move the population six steps forward in time, capturing the transients by storing the vector of abundance in each size bin at each time-step:

```
storage <- matrix(NA,length(Pmat[,1]),6)
n.new.dist <- n.new.dist0
for (k in 1:length(storage[1,])) {
   n.new.dist <- IPM%*%n.new.dist
         storage[,k] <- n.new.dist
 }
```

We plot the P matrix, F matrix, the starting size distribution, and the size distribution in the first six time steps 7.3:

```
# make a plot of the current IPM
par(mfrow=c(2,2))
image(Pmat@meshpoints,Pmat@meshpoints,
         t(Pmat), xlab="Size", ylab="Size at t+1")
contour(Pmat@meshpoints,Pmat@meshpoints,
         t(Pmat), add=TRUE)
image(Pmat@meshpoints,Pmat@meshpoints,
         t(Fmat), xlab="Size", ylab="Size at t+1")
contour(Pmat@meshpoints,Pmat@meshpoints,
         t(Fmat), add=TRUE)
# show the starting distribution
hist(startingSizes, xlab="starting sizes", ylab="",
      main="",col="grey")
#add a figure of this movement into the future
image(1:ncol(storage),Pmat@meshpoints,t(log(storage)),
       xlab="time", ylab="Size")
abline(v=1.5+(1*c(0:10)))
```

Obviously, these dynamics would be more interesting if the IPMs varied from year to year, or density dependence was operating. Exactly the same principles would apply, except that we would multiply the population vector by a different IPM from one year to the next, possibly recalculated

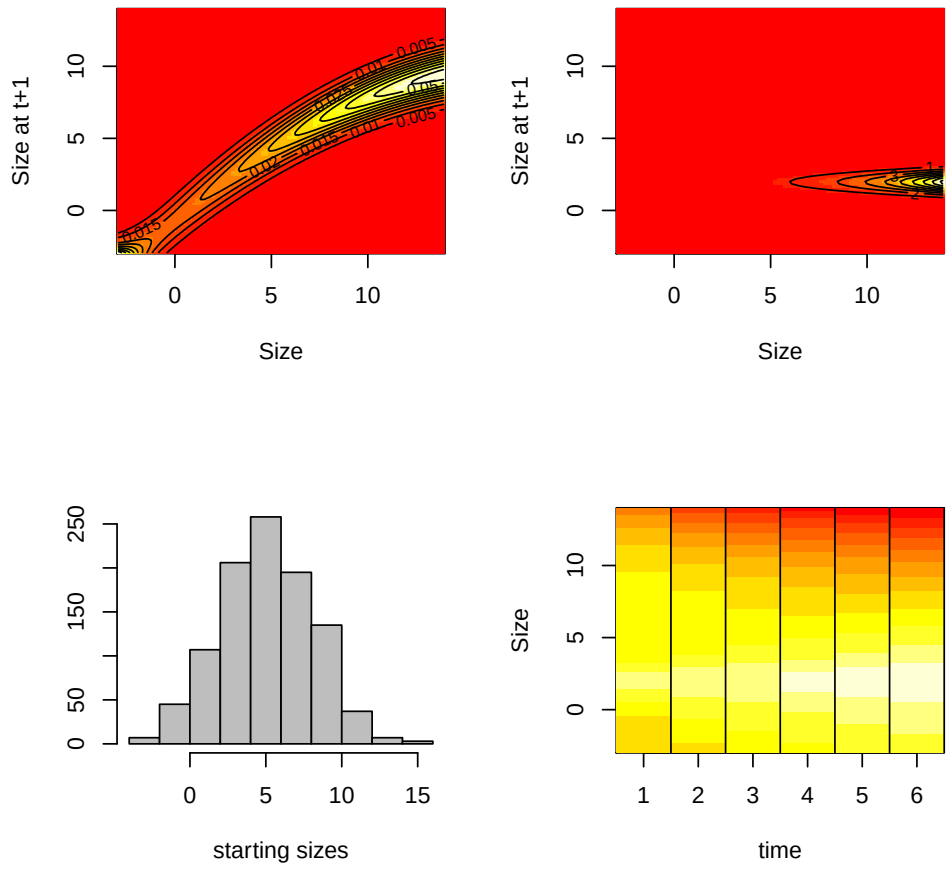

Figure 5.1: Components of an analysis of transient dynamics in a constant environment: IPM subkernels in the top row (Pmatrix, then Fmatrix), and the starting size distribution (left) and its iterations (log scale, right) in the lower row.

to incorporate density dependence. It is also potentially useful to examine what happens following multiplication of just the P matrix (instead of the full IPM); i.e., explore the cohort dynamics in terms of changes in population structure purely due to growth and survival.

5.4.2 Stochastic population growth rates

If demographic rates vary over time, following a stationary distribution of environments the population growth rate and stable population structure will converge to constant distributions via the weak ergodic theorem (Tuljapurkar, 1990). This population growth rate and population structure will differ importantly from what is predicted under a constant environment. Here, we illustrate how to estimate the stochastic population rate of increase, and some of its characteristics.

As in the previous section, we generate some data, and create growth, survival, and fertility objects:

```
dff <- generateData()
gr1 <- makeGrowthObj(dataf = dff,
                     Formula=sizeNext~size,
                     regType="constantVar",
                     Family="gaussian")
sv1 <- makeSurvObj(dff, Formula = surv~size+size2)
fv1 <- makeFecObj(dataf=dff,Formula=list(fec~size), Transform="log")
```

We then reduce the magnitude of the intercept and slope terms in the size-dependent fecundity function, relative to the default, since otherwise results are rather uninteresting (keeping in mind that fecundity is modeled on a log scale, so a negative intercept is fine):

```
fv1@fitFec[[1]]$coefficients <- c(-1,0.35)
```

Now imagine that there is another kind of year where fertility and growth are the same, but survival is poor. We can define a model that does this by coercing our survival object to have different (smaller) parameters.

```
# create poor survival object
sv2 <- coerceSurvObj(survObj=sv1,coeff=c(-3,0,0))
```

We can create two IPMs reflecting these two different types of environments, one with `sv1`, corresponding to "good" years, and one with `sv2`, corresponding to "bad" years; fertility and growth remain the same across years.

```
Pmat1 <- makeIPMPmatrix(nBigMatrix=50,minSize = -2,maxSize = 12,
         growObj = gr1, survObj = sv1)
Pmat2 <- makeIPMPmatrix(nBigMatrix=50,minSize = -2,maxSize = 12,
         growObj = gr1, survObj = sv2)
Fmat <- makeIPMFmatrix(nBigMatrix=50,minSize = -2,maxSize = 12,
         fecObj =fv1)
IPM1 <- Pmat1+Fmat  ##good year
IPM2 <- Pmat2+Fmat  ##bad year
```

We compare their rates of increase λ in a constant environment:

```
 Re(eigen(IPM1)$value[1])
[1] 1.103917
 Re(eigen(IPM2)$value[1])
[1] 0.8088856
```

as well as the arithmetic mean growth rate of the two:

```
 0.5*(Re(eigen(IPM1)$value[1])+Re(eigen(IPM2)$value[1]))
[1] 0.9564011
```

We can also compare the stochastic population growth rate λ_s of a population experiencing both "good" years (corresponding to `sv1`) and "bad" years (corresponding to `sv2`) against the stochastic population growth rates of populations where yearly demography is defined only by "good" vs. "bad" years.

```
 #the two year-types
 exp(stochGrowthRateSampleList(list(IPM1,IPM2),nRunIn=1000,tMax=5000))
[1] 0.9411612
 #only the good years (should match the above)
 exp(stochGrowthRateSampleList(list(IPM1,IPM1),nRunIn=1000,tMax=5000))
[1] 1.103917
 #only the bad years (should match the above)
 exp(stochGrowthRateSampleList(list(IPM2,IPM2),nRunIn=1000,tMax=5000))
[1] 0.8088856
```

Note that the function `stochGrowthRateSampleList` returns the expectation of the log of the growth rate (hence the value is exponentiated in the above in order to compare it to the eigenvalues above).

The simple expectation of the growth rate (calculated above by taking the average of the two constant environment growth rates) will tend to over-estimate the stochastic population growth rate (i.e., this value is greater than the value estimated with `stochGrowthRateSampleList`) due to Jensen's inequality. To illustrate why this occurs, we can plot a few sample time-series for situations where years are either evenly split between good and bad, vs. only good or only bad.

```
 IPMlist <- list(IPM1,IPM2)
 # initiate population with 10 individuals
 # at the stable pop structure for the first environment
 nt0 <-abs(Re(eigen(IPM1)$vector[,1])*10)
 # create vectors fos storage
 rt <- rtGood <- rtBad <- matrix(NA,20,100)
 #run 20 simulations
 for (j in 1:20) {
          ntGood <- ntBad <- nt <- nt0
           #loop over 100 years
```

```
        for (k in 1:100) {
                #pick a year type at random and iterate the pop
                yrtype <- sample(1:2,size=1)
                nt <- IPMlist[[yrtype]]%*%nt
                rt[j,k] <- sum(nt)
                #store what happens in only good and only bad environments
                ntGood <- IPM1%*%ntGood
                rtGood[j,k] <- sum(ntGood)
                ntBad <- IPM2%*%ntBad
                rtBad[j,k] <- sum(ntBad)
 }}
```

Plotting this shows that the population in the variable environment is intermediate in size, but less than the arithmetic mean of the two populations, as expected under Jensen's inequality (Fig. 7.1).

```
par(mfrow=c(1,3),bty="l")
par(mfrow=c(1,1),bty="l")
matplot(t(rt),type="l",xlab="time (yrs)", ylab="total population size",
        log="y", ylim=range(c(rtGood,rtBad)), lty=1)
# we can check to see that all the rtGoods and all the rtBads
# are the same at every time-step
# so we are just plotting the first simulation
points(rtGood[1,], type="l",lwd=2,lty=2)
points(rtBad[1,], type="l",lwd=2,lty=2)
```

5.4.3 Creating compound matrices

Another way of exploring stochastic environments is via the mega-matrix, which represents movement through a markovian sequence of environments using a block matrix structure. An example of this type of analysis is in Metcalf et al. (2009), where data from Clark & Clark were used to analyse growth and survival of tropical trees at La Selva, Costa Rica (see also Yule et al. 2013). We supply an IPMpack-organized version of the most recently available data (see: http://esapubs.org/archive/ecol/E093/019/metadata.htm); this data set has more years of data than those used in Metcalf et al. (2009). The function to accomplish this organization is found at the end of this appendix (Supplementary Function). First we load the data (changing the path in filename to whatever is appropriate on your computer):

```
dff <- read.csv("cleanedClarkClark.csv")
```

and have a look at the data:

```
head(dff)
```

```
  X  size sizeNext crown crownNext incr surv exactDate
1 1  28.0     25.3     2         4   NA    1  5/9/1983
2 3 278.0    276.0     4         4 -1.9    1 5/12/1983
```

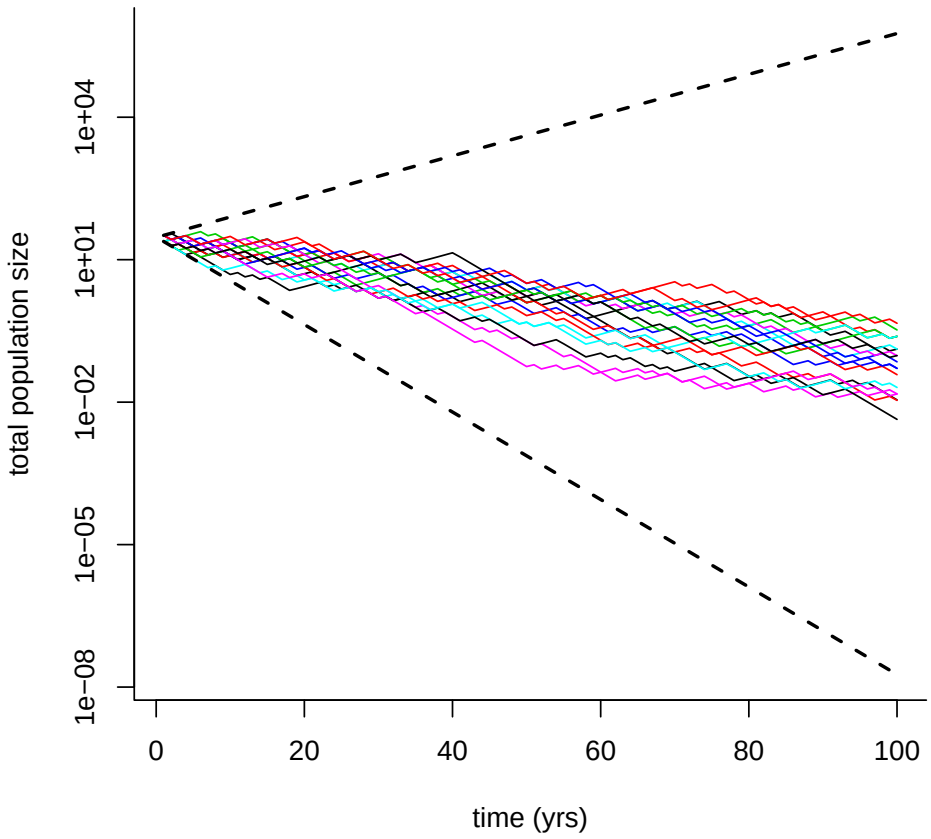

Figure 5.2: Some sample time-series: the upper and lower lines show population size in only the good or only the bad environment respectively.

```
3 4  36.4     37.8     2         3  1.4    1 5/12/1983
4 5   6.4      5.7     3         2   NA    1 5/12/1983
5 6 827.0       NA     5         6   NA   NA 5/12/1983
6 7   6.5      4.8     2         1   NA    1 5/12/1983
  exactDatel species year
1  5/21/1984      Dp 1983
2  5/21/1984      Dp 1983
3  5/21/1984      Dp 1983
4  5/21/1984      Dp 1983
5  5/21/1984      Dp 1983
6  5/21/1984      Dp 1983
```

We select a species, for example, one of the pioneer species, *Cecropia obtusifolia* (a full list of the two letter species codes is available on the Ecological Archives website):

```
dff <- dff[dff$species=="Co",]
```

We then whittle down the data-set to get rid of NAs with respect to size and current light environment (we do not get rids of NAs in sizeNext or crownNext, since this would eliminate our mortality data):

```
dff <- dff[!is.na(dff$size) & !is.na(dff$crown),]
```

Now we put size and sizeNext on a log scale:

```
dff$size <- log(dff$size)
dff$sizeNext <- log(dff$sizeNext)
```

and we construct growth and survival objects. Let us build simple ones to start with:

```
gr1 <- makeGrowthObj(dataf=dff, Formula = sizeNext ~ size+size2)
sv1 <- makeSurvObj(dataf=dff, Formula = surv ~ size+size2)
```

Note that there are many reasons why `gr1` might not be a very good model for tree growth, especially the fact that it is possible for trees to shrink (Zuidema et al. 2010); but we will retain this here for simplicity. We can now build the corresponding P matrix:

```
Pmat <- makeIPMPmatrix(minSize = 0.5, maxSize = 8, nBigMatrix=200,
                           growObj=gr1, survObj=sv1,
                           integrateType = "midpoint", correction="constant")
```

and plot it using `image`:

```
image(Pmat@meshpoints, Pmat@meshpoints, t(log(Pmat)),
       xlab = "Size at t (mm)",
       ylab = "Size at t+1 (mm)", axes=FALSE)
#add axis on the original scale (rather than log)
axis(1,at=log(c(5,10,50,100,500,1000)),lab=c(5,10,50,100,500,1000))
axis(2,at=log(c(15,0,50,100,500,1000)),lab=c(5,10,50,100,500,1000))
#add contours(note, they are on the log scale...)
contour(Pmat@meshpoints, Pmat@meshpoints, t(log(Pmat)), add=TRUE)
```

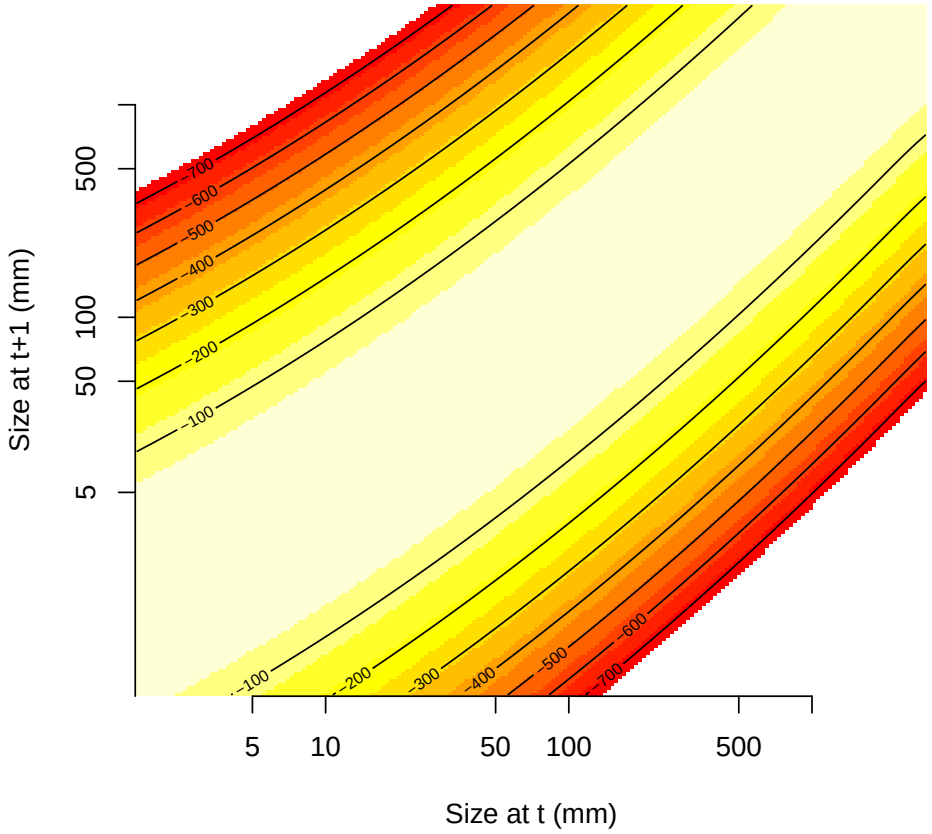

Figure 5.3: Pmatrix (log scale)

Ideally extensive tests should be performed for adequacy of the number of bins, size range, etc, of this IPM (see Appendix A, `convergeIPM`, and related functions), in addition to tests of the functional form and predictors, but here we will retain these dimensions and structure, just to demonstrate the construction of a megamatrix.

We now can estimate the first passage time to a chosen size for this model and plot it. We choose 30cm, since that is approximately the diameter at which many species reach the canopy:

```
pTime <- passageTime(log(300),Pmat)
LE <- meanLifeExpect(Pmat)
par(mfrow=c(1,2),bty="l")
plot(Pmat@meshpoints,pTime,type="l",
      xlab="Size (log scale, mm)",ylab="Passage time to 30cm (years)",
      log="y", xlim=c(0.5,0.95*log(300)))
plot(Pmat@meshpoints,LE,type="l",
      xlab="Size (log scale, mm)",ylab="Life expectancy",
      log="y", xlim=range(dff$size,na.rm=TRUE))
```

The x range of this plot has been chosen to ignore the probability of achieving 30cm for sizes larger than 30cm, since we are interested in first passage time. Now let us build a Pmatrix including discrete covariates. IPMpack is most comfortable with discrete covariates (treated as factors) and building compound matrices if the discrete covariates are named `covariate` and `covariateNext`:

```
dff$covariate <- dff$crown
dff$covariateNext <- dff$crownNext
```

Now we can build the corresponding, more complicated growth and survival objects:

```
gr2 <- makeGrowthObj(dataf=dff,
                     Formula = sizeNext ~ size+size2+covariate)
sv2 <- makeSurvObj(dataf=dff,
                   Formula = surv ~ size+size2)
```

We also need a description of how individuals are moving between different light environments:

```
env1 <- makeEnvObj(dff[!is.na(dff$covariateNext),])
env1
```

```
An object of class "envMatrix"
           [,1]       [,2]       [,3]       [,4]
[1,] 0.39024390 0.02352941 0.00000000 0.01086957
[2,] 0.41463415 0.38823529 0.10784314 0.00000000
[3,] 0.12195122 0.43529412 0.70588235 0.15217391
[4,] 0.02439024 0.07058824 0.12745098 0.41304348
[5,] 0.04878049 0.08235294 0.05882353 0.42391304
[6,] 0.00000000 0.00000000 0.00000000 0.00000000
            [,5]      [,6]
[1,] 0.000000000 0.0000000
```

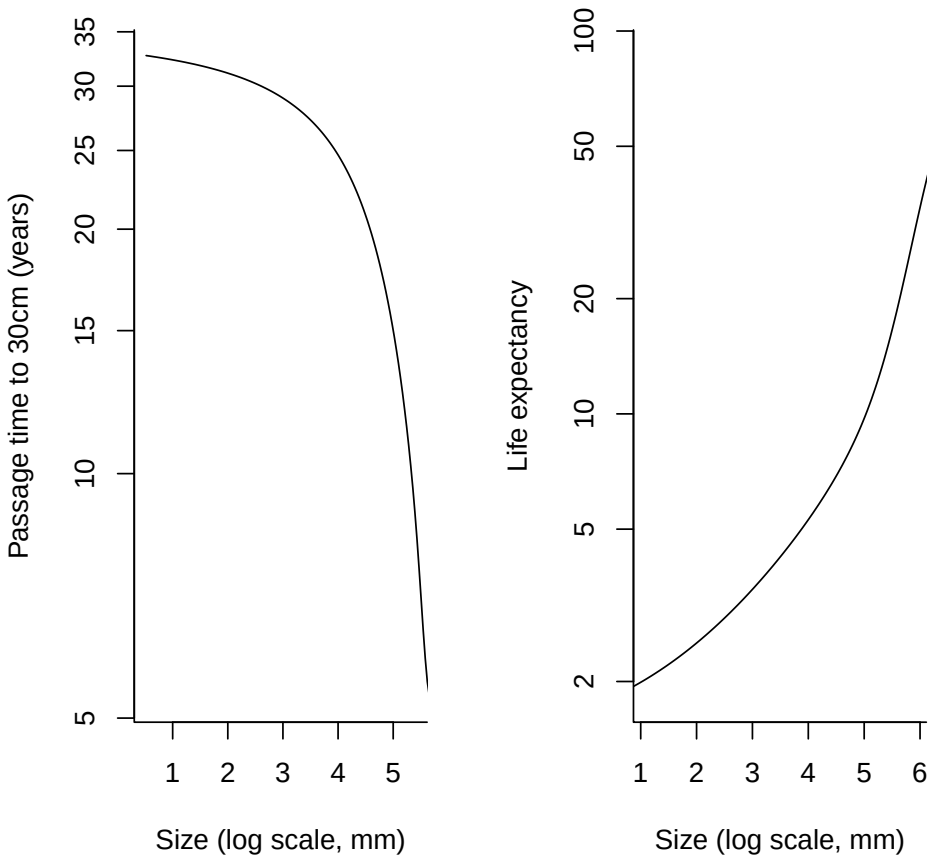

Figure 5.4: Passage time and life expectancy, ignoring light environment

```
[2,] 0.001160093 0.0000000
[3,] 0.003480278 0.0000000
[4,] 0.044083527 0.0000000
[5,] 0.941995360 0.7272727
[6,] 0.009280742 0.2727273
Slot "nEnvClass":
[1] 6
```

This defines the Markov chain for the light environment. There are six rows and columns in this matrix since there are six possible light environments. The transition probabilities in this matrix include not only gap formation and closure (determined by the neighboring trees), but also the ability of individuals to occupy and traverse the vertical strata of the forest. The next step is to construct a mega-matrix that combines environmental and demographic transitions:

```
PmatComp <- makeCompoundPmatrix(nEnvClass =6,
                                envMatrix=env1,minSize = 0.5,
                                maxSize = 8, nBigMatrix=200,
                                growObj=gr2, survObj=sv2,
                                integrateType = "midpoint",
                                correction="constant")
```

The megamatrix is at the heart of closed-form equations that quantify remaining life expectancy and age-specific survivorship in a Markovian environment, conditional on initial environment. We can have look at the megamatrix using the code below (figure not shown here only because it turns out to be rather large):

```
image(1:nrow(PmatComp), 1:ncol(PmatComp), t(log(PmatComp)),
       xlab = "Size at t (log scale, mm)",
       ylab = "Size at t+1 (log scale, mm)", axes = FALSE)
axis(1, at = 1:nrow(PmatComp),
      lab = round(rep(PmatComp@meshpoints,PmatComp@nEnvClass), 2))
axis(2, at = 1:nrow(PmatComp),
      lab = round(rep(PmatComp@meshpoints,PmatComp@nEnvClass), 2))
abline(h = length(PmatComp@meshpoints) * (1:PmatComp@nEnvClass))
abline(v = length(PmatComp@meshpoints) * (1:PmatComp@nEnvClass))
```

Tuljapurkar & Horvitz (2006) have defined relationships that allow the estimation of key quantities, such as stochastic life expectancy, for individuals born at particular sizes in particular environments, experiencing an environment changing according to a Markov process, as defined by a mega-matrix.

```
LEComp<-.stochLifeExpect(IPMmatrix=PmatComp,envMatrix=env1)
```

We can then plot stochastic life expectancy:

```
#create template for plotting
par(mfrow=c(1,1),bty="l")
plot(PmatComp@meshpoints,LEComp[1,],
```

```
         ylab = "Life expectancy (years)", xlab = "Size (mm)",
         type = "n", col = "dark gray", log="y",
         ylim = c(1, max(LEComp)),
         xlim=range(dff$size,na.rm=TRUE),axes=FALSE)
 #add constant env estimate
 points(Pmat@meshpoints,LE,type="l",lwd=3,lty=1, col="grey")
 #add axes (to show size on the original scale, rather than logged)
 axis(2)
 axis(1,at=log(c(1,5,10,20,30,50,100,200,400,500,1000)),
       lab=c(1,5,10,20,30,50,100,200,400,500,1000))
 #loop over the light environments
 cols <- rev(rainbow(n=6,start=0,end=2/5))
 for (j in 1:nrow(LEComp)) {
         points(PmatComp@meshpoints,LEComp[j,], type="l",col=cols[j])
  }
```

Note that this is still a hidden function in IPMpack, since it still requires some checking; so use with care. We will use this function to compare projected passage time for individuals starting in the different light environments.

```
 pTimeComp <- stochPassageTime(log(300),PmatComp, env1)
```

and now we can plot this:

```
 #create template for plotting
 par(mfrow=c(1,1),bty="l")
 plot(PmatComp@meshpoints,pTimeComp[1:length(PmatComp@meshpoints)],
         ylab = "Passage time (years)", xlab = "Size (mm)",
         type = "n", col = "dark gray", log="y",
         xlim=c(PmatComp@meshpoints[1],0.95*log(300)),
         ylim = c(5, max(pTime)), axes=FALSE)
 #add axes (to show size on the original scale, rather than logged)
 axis(2)
 axis(1,at=log(c(1,5,10,20,30,50,100,200,400,500,1000)),
       lab=c(1,5,10,20,30,50,100,200,400,500,1000))
 #add the line from the average environment
 points(Pmat@meshpoints,pTime,type="l",lwd=3,lty=1, col="grey")
 #loop over the light environments
 cols <- rev(rainbow(n=6,start=0,end=2/5))
 for (j in 1:nrow(env1)) {
      points(PmatComp@meshpoints,
             pTimeComp[((j-1)*length(PmatComp@meshpoints)+1):
                       (j*length(PmatComp@meshpoints))],
```

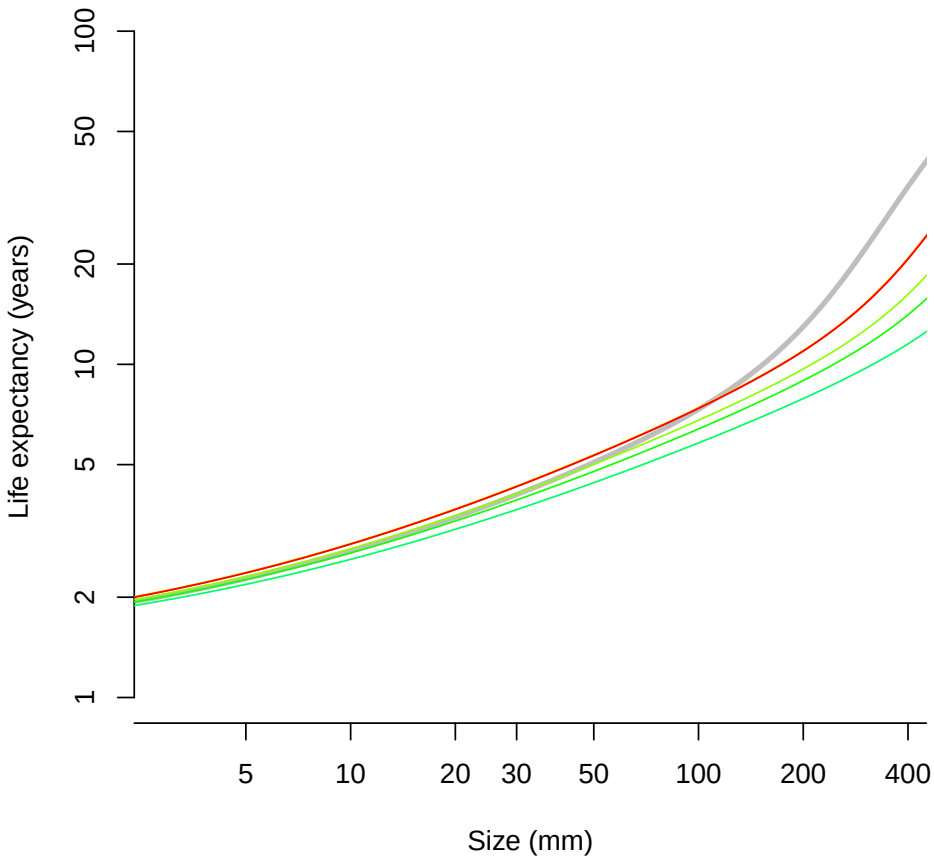

Figure 5.5: Life expectancy for cohorts recruiting into different environments; constant environment prediction shown in grey

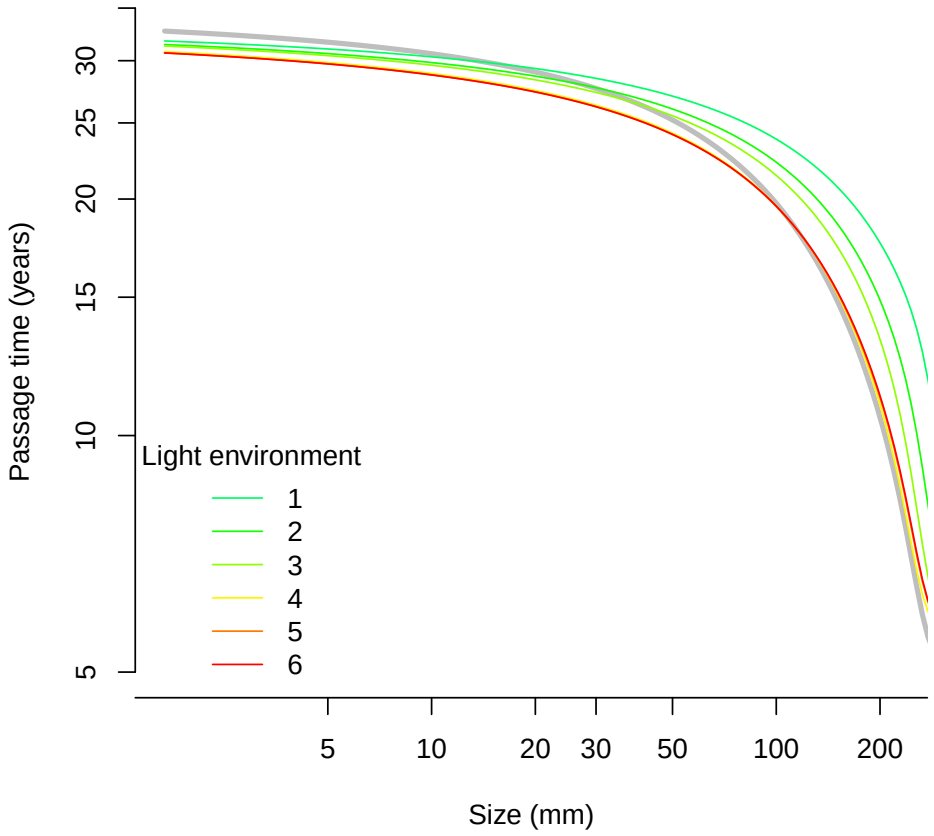

Figure 5.6: Passage time to 30cm (years), with light environment fitted; grey line shows the constant environment prediction

```
            type="l",col=cols[j])
 }
#put in a legend
legend("bottomleft",legend=1:nrow(env1),
        col=cols[1:nrow(env1)],lty=1,
        bty="n",title="Light environment")
```

This calculation is simply the extension of the constant environment model. This figure indicates that individuals of this species of the same size that recruit into a higher light environment (as measured by the light index) tend to reach 30cm faster, which is a highly plausible conclusion. However, the exact values shown here are highly unlikely to be very accurate, given the absence of model-checking.

5.5 References

- Caswell, H. 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer. Massachusetts, USA.
- Caswell, H. 2007. Sensitivity analysis of transient population dynamics. Ecology Letters 10, 1-15.
- Haridas, C.V. & Tuljapurkar, S. 2007. Time, transients and elasticity: Ecology Letters 10, 1143-1153.
- Metcalf, C.J.E., Horvitz, C., Tuljapurkar S. & Clark, D.A. 2009. A time to grow and a time to die: a new way to analyze the dynamics of size, light, age and death of tropical trees. Ecology 90, 2766-2778.
- Ozgul, A., Coulson, T., Reynolds, A., Cameron, T.C. & Benton, T.G. (2012). Population Responses to Perturbations: The Importance of Trait-Based Analysis Illustrated through a Microcosm Experiment. The American Naturalist, 179, 582âĂŞ594.
- Tuljapurkar, S. 1990. Population Dynamics in Variable Environments. Springer. New York, USA.
- Tuljapurkar, S., & C. Horvitz. 2006. From stage to age in variable environments: life expectancy and survivorship. Ecology 87, 1497âĂŞ1509.
- Vindenes, Y., Soether, B.E. & Engen, S. (2011). Effects of demographic structure on key properties of stochastic density-independent population dynamics. Theoretical Population Biology, 82, 253âĂŞ263.
- Yule, K.M., Miller, T.E.X. & Rudgers, J.A. (2013). Costs, benefits, and loss of vertically transmitted symbionts affect host population dynamics. Oikos, in press.
- Zuidema, P.A., Jongejans, E., Chien, P.D., During, H.J., & Schieving, F. 2010. Integral Projection Models for trees: a new parameterization and a validation of model output. Journal of Ecology 98, 345-355.

5.6 Supplementary function

```
## Function pull in the data taken from
#  http://esapubs.org/archive/ecol/E093/019/metadata.htm
## Ecological Archives,
## and organize it into a format recognized by IPMpack
#
# parameters - filename - file describing location of the data
```

```
#            - species - chosen species  - two letter code, e.g.,
#                  Be  Ci  Co  Dp  Ha  Hm  La  Mg  Pm  Sa or "all"
#              species codes are described in the online documentation
#              for this data on ecological archives
#
# returns - a data-frame with headings size, sizenext (adjusted for time interval),
#                incr (raw increment), surv (survival), spcode,
#              and exactDate and exactDatel in days
#
getData <- function(filename="data/",
                    species="all"){

     #bring in the data
     dfDiam <- read.delim(paste(filename,"Diameter_1983_2010.txt", sep=""),
                          header=TRUE)
     if (species!="all") dfDiam <- dfDiam[dfDiam$species==species,]
     spcode <- dfDiam[,2]
     dfDiam <- as.matrix(dfDiam[,3:30])
     dfDiam[dfDiam==-999] <- NA
     #increment
     dfincr <- read.delim(paste(filename,"Diameter_growth_1983_2010.txt", sep=""),
                          header=TRUE)
     if (species!="all") dfincr <- dfincr[dfincr$sp==species,]
     dfincr <- as.matrix(dfincr[,3:29])
     dfincr[dfincr==-999] <- NA
     #exact dates
     dfexactDate <- read.delim(paste(filename,"Dates_1983_2010.txt", sep=""),
                               header=TRUE)
     if (species!="all") dfexactDate <- dfexactDate[dfexactDate$sp==species,]
     dfexactDate <- as.matrix(dfexactDate[,3:30])
     #bring in the crown position data
     dfCrown <- read.delim(paste(filename,"Crown_positions_1983_2010.txt", sep=""),
                           header=TRUE)
     #they are aligned - cbind(dfCrown$id,dfDiam$id)
     if (species!="all") dfCrown <- dfCrown[dfCrown$species==species,]
     dfCrown <- as.matrix(dfCrown[,3:30])
     dfCrown[dfCrown==-999] <- NA

     ##change crownpos to discrete index as used in Metcalf et al. 2009
```

```
### (modify this if you would like more / less precision, or
### to return the original values)
dfCrown[dfCrown==5]<-6
dfCrown[dfCrown==4 | dfCrown==4.5]<-5
dfCrown[dfCrown==3.25 |dfCrown==3 | dfCrown==3.5]<-4
dfCrown[dfCrown==2.5 | dfCrown==2.75]<-3
dfCrown[dfCrown==2.25]<-2
dfCrown[dfCrown==1.5 | dfCrown==1.75 |
        dfCrown==1.25 | dfCrown==1.55 | dfCrown==1.7]<-1

#they are aligned... so dont need to worry about that
#cbind(dfDiam[,1:2],dfincr[,1:2],dfexactDate[,1:2])

#make survival - note needs to be lagged one back
surv <- matrix(NA,nrow(dfDiam),ncol(dfDiam))
surv[which(dfDiam<9002,arr.ind=TRUE)] <- 1
surv[dfDiam>9001] <- 0
surv <- cbind(surv[,2:ncol(surv)],rep(NA,nrow(surv)))

#clean up dfDiam
dfDiam[dfDiam>8999] <- NA
dfincr[dfincr>8999] <- NA

#create a data-frame
dataf <- data.frame(size=c(dfDiam[,1:(ncol(dfDiam)-1)]),
                    sizeNext=c(dfDiam[,2:ncol(dfDiam)]),
                    crown=c(dfCrown[,1:(ncol(dfDiam)-1)]),
                    crownNext=c(dfCrown[,2:ncol(dfDiam)]),
                    incr=c(dfincr[,1:(ncol(dfDiam)-1)]),
                    surv= c(surv[,1:(ncol(dfDiam)-1)]),
                    exactDate=c(dfexactDate[,1:(ncol(dfDiam)-1)]),
                    exactDatel=c(dfexactDate[,2:ncol(dfDiam)]),
                    species=rep(spcode,ncol(dfDiam)-1))

dataf$year <- as.numeric(substring(as.character(dataf$exactDate),
                                   nchar(as.character(dataf$exactDate))-3,
                                   nchar(as.character(dataf$exactDate))))

#get rid of uncessary rows where no size measurements are available
```

```
    dataf <- dataf[!(is.na(dataf$size)),]

    return(dataf)
}
```

Chapter 6

Appendix F: A niche model for *Actaea spicata*

Contact: Johan P. Dahlgren (Johan.Dahlgren@botan.su.se)

Abstract

Here we give an example of how IPMs can be used to predict the geographic distribution of a species' niche, under a demographic definition of the niche - i.e., those conditions where population sizes can be maintained or increased. The example is based on an environment-dependent IPM for the herb *Actaea spicata* published by Dahlgren and Ehrlen (2009, 2011), and additional spatial information on environmental variation in the study area. We provide simulated data for readers who wish to practice the process of IPM parameterization using regressions, as well as the R code used to calculate population growth rates.

6.1 Introduction

For many ecological applications, it is critical to distinguish the causes of differences in dynamics among (sub-)populations. IPMs are well suited for this because covariates, for example, describing environmental conditions, can easily be included in the regression models describing vital rates. Random effects can also be included, to capture differences due to unmeasured causes. In this case study, soil potassium concentration affects the growth rate of individuals, and consequently population dynamics. When covariates that differentiate populations can be measured at the landscape level (here, using Geographic Information System layers) the IPMs can be projected across the landscape to predict the spatial distribution of population dynamics, which in turn can be used to understand range dynamics, metapopulation dynamics, niche distributions, regional population viability, etc.

6.2 Study system/objectives

We illustrate the construction of an IPM for the forest herb *Actaea spicata* (Ranunculaceae), which is affected by forest composition via soil potassium concentration (Dahlgren and Ehrlen 2009, 2011). We then use this model to determine the spatial distribution of *Actaea's* niche (defined in demographic terms) based on information on forest composition in the study area. *Actaea spicata* is a long-lived, Eurasian, non-clonal herb of nutrient-rich forests that are composed of a mixture of deciduous broadleaf species and spruce. *Actaea* populations were studied in eastern Sweden (2004-2006 in Tullgarn Natural Reserve), where this species is restricted to mid-successional forests in which the majority of deciduous trees have not yet been replaced by the dominant, late-successional species Norway spruce (*Picea abies*). Growth of *Actaea* individuals is affected by soil potassium concentration, which is in turn related to the proportion of spruce trees in the immediate area. Soil potassium declines as spruce takes over because the nutrient input provided by deciduous leaves declines. Given approximate data on spruce proportion across the study area, we use the IPM to produce a map of *Actaea's* expected population growth rate; areas with a predicted population growth rate (λ) greater than 1 are considered suitable habitat for this species (Fig. 6.4a). Further, we used the information in Dahlgren and Ehrlen (2011) to infer the number of years since spruce colonization from the current proportion of spruce. This allowed us to forecast the geographic distribution of *Actaea's* suitable habitat 25 years into the future (Fig. 6.4b).

6.3 Data

For proprietary reasons, we provide simulated data for *A. spicata*, rather than the original data. This allows us to obtain parameter values similar to those obtained from the real data and proceed with the analysis. The first step is to load the data.

```
(load(ActaeaRegDat.RData))
```

```
[1] "ActaeaRegDat"
```

```
head(ActaeaRegDat)
```

	x	y	K	surv	flow	fruits
1	4.592534	8.866817	32.23421	1	0	NA
2	1.032709	NA	30.95493	0	0	NA
3	7.594417	7.381519	35.17560	1	0	NA
4	1.206751	NA	31.00796	0	0	NA
5	1.858199	2.910622	32.01475	1	0	NA
6	13.309052	12.178099	31.48942	1	1	74

The simulated data include the columns x (size in year t), y (size in year t+1), K (soil potassium concentration), surv (survival), flow (flowering) and fruits (number of fruits). Size-dependent vital rates are illustrated in Fig. 6.1.

```
par(mfrow=c(2,2),mar=c(4,4,2,1))
plot(ActaeaRegDat$x,jitter(ActaeaRegDat$surv),xlab="Size (t)",
   ylab="Survival to t+1") # jittered
plot(ActaeaRegDat$x,ActaeaRegDat$y,xlab="Size (t)",ylab="Size (t+1)")
plot(ActaeaRegDat$x,jitter(ActaeaRegDat$flow),xlab="Size (t)",
   ylab="Flowering probability") # jittered
plot(ActaeaRegDat$x,ActaeaRegDat$fruits,xlab="Size (t)",
   ylab="Fruit Number")
```

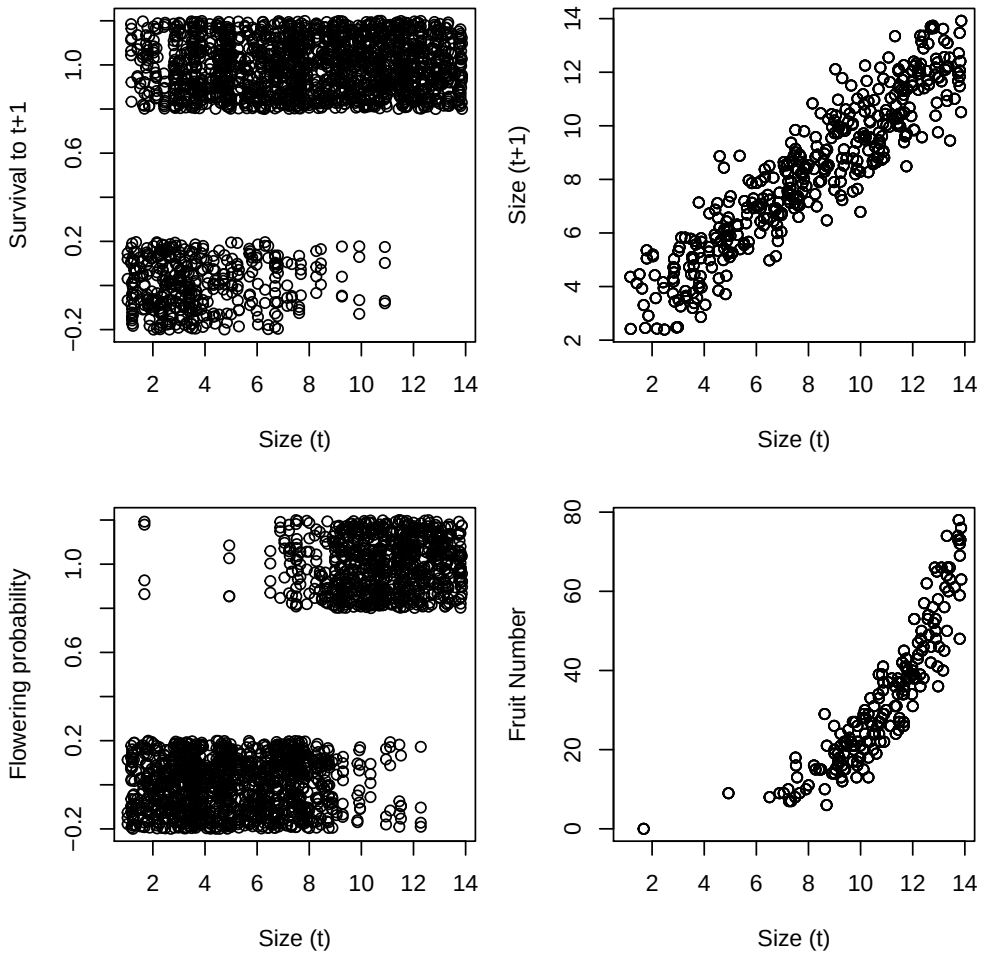

Figure 6.1: Exploration of vital rate data.

6.4 Analysis

The Actaea IPM is presented in more detail in Dahlgren and Ehrlen (2009). It is similar to the IPM in Appendix A, except that (1) flowering probability and the number of fruits are modeled as two separate processes, captured by two regression models, and (2) seeds and seedlings are modeled as separate classes. Three coupled equations model seeds (B) and seedlings (S) as discrete classes and the adult population (n) as a continuous distribution:

$$B_{t+1} = \int_{\Omega} p_{flower}(z)\ f_{fruit}(z)\ f_{seeds/fruit}\ n_t(z)\ dz \tag{6.1}$$

$$S_{t+1} = p_{estab}\ B_t \tag{6.2}$$

$$n_{t+1}(z') = s_{seed\ surv}\ f_{recruit\ size}(z')\ S_t\ +\ \int_{\Omega} s(z)\ g(z',z)\ n_t(z)\ dz \tag{6.3}$$

The state variable is size, calculated as the sum of squared stem diameters times total plant height. The model includes two covariates - one describing the dependence of individual growth on potassium concentration, the other describing the effect of seed predation (we assume that 20% of fruits are lost to seed predation at all locations). Individuals were arranged in plots within sites; these hierarchies were accounted for by including plot and site as nested random factors in the original vital rate regressions, but this information is suppressed here to simplify the analysis. Other studies including multiple discrete states to capture differences in early life stages (e.g. seedlings of various ages) include Williams et al. (2010),

6.4.1 Build regressions for vital rate functions

Below we provide estimates for most model parameters from the original data. However, survival, growth, and fecundity parameters can be estimated from the simulated data using the following regressions. Growth is modeled as a function of size and soil potassium concentration, while other vital rates depend only on size (Fig. 6.3).

```
 # Growth: linear regression
 summary(grow.mod<-lm(y~x+K,data=ActaeaRegDat))
Call:
lm(formula = y ~ x + K, data = ActaeaRegDat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.97884 -0.75005  0.01853  0.70411  3.15174

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.584787   0.270898   5.850 5.94e-09 ***
x           0.750455   0.008002  93.779  < 2e-16 ***
K           0.021565   0.008782   2.455   0.0142 *
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

```
Residual standard error: 1.071 on 1601 degrees of freedom
  (396 observations deleted due to missingness)
Multiple R-squared:  0.8472,        Adjusted R-squared:  0.847
F-statistic:  4438 on 2 and 1601 DF,  p-value: < 2.2e-16
```

Note that growth is modeled as a multiple regression with size in year t + 1 dependent upon size in year t and soil potassium. Thus, it is only via individual growth that the population is affected by forest composition (via soil potassium).

```
 # Survival: logistic regression
 summary(surv.mod<-glm(surv~x,family=binomial,data=ActaeaRegDat))
Call:
glm(formula = surv ~ x, family = binomial, data = ActaeaRegDat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8282   0.1161   0.2575   0.5848   1.5459

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.41020    0.14226  -9.913   <2e-16 ***
x            0.49450    0.02789  17.733   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1990.5  on 1999  degrees of freedom
Residual deviance: 1445.1  on 1998  degrees of freedom
AIC: 1449.1

Number of Fisher Scoring iterations: 6
 # Flowering: logistic regression
 summary(flow.mod<-glm(flow~x,family=binomial,data=ActaeaRegDat))
Call:
glm(formula = flow ~ x, family = binomial, data = ActaeaRegDat)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9071  -0.2299  -0.0522   0.2432   3.9704
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.79455    0.47189  -20.76   <2e-16 ***
x            1.14052    0.05437   20.98   <2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2674.99  on 1999  degrees of freedom
Residual deviance:  903.29  on 1998  degrees of freedom
AIC: 907.29

Number of Fisher Scoring iterations: 7
 # Fruit number: poisson regression
 summary(fruit.mod<-glm(fruits~x,family=poisson,data=ActaeaRegDat))
Call:
glm(formula = fruits ~ x, family = poisson, data = ActaeaRegDat)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.78434  -0.76446   0.00307   0.70894   3.12030

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.235928   0.046706   5.051 4.39e-07 ***
x           0.288904   0.003959  72.976  < 2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 6848.47  on 779  degrees of freedom
Residual deviance:  874.64  on 778  degrees of freedom
  (1220 observations deleted due to missingness)
AIC: 4919.3

Number of Fisher Scoring iterations: 4
```

The fit of these regressions can be inspected by adding the predicted curves to the scatter plots of data in Fig. 6.1 above, as follows in Fig. 6.2.

```
par(mfrow=c(2,2),mar=c(4,4,2,1))
xx<-seq(2,14,length.out=100) # range of sizes to plot the curves for
plot(ActaeaRegDat$x,jitter(ActaeaRegDat$surv),
   xlab="Size (t)", ylab="Survival to t+1") # jittered
lines(xx,predict(surv.mod,data.frame(x=xx),type=response),
   col=red,lwd=3)
plot(ActaeaRegDat$x,ActaeaRegDat$y,xlab="Size (t)",
   ylab="Size (t+1)")
lines(xx,predict(grow.mod,data.frame(x=xx,K=25),type=response),
   col=red,lwd=3) # at mean soil potassium concentration
plot(ActaeaRegDat$x,jitter(ActaeaRegDat$flow),
   xlab="Size (t)",ylab="Flowering probability")
lines(xx,predict(flow.mod,data.frame(x=xx),type=response),
   col=red,lwd=3)
plot(ActaeaRegDat$x,ActaeaRegDat$fruits,xlab="Size (t)",
   ylab="Fruit Number")
lines(xx,predict(fruit.mod,data.frame(x=xx),type=response),
   col=red,lwd=3)
```

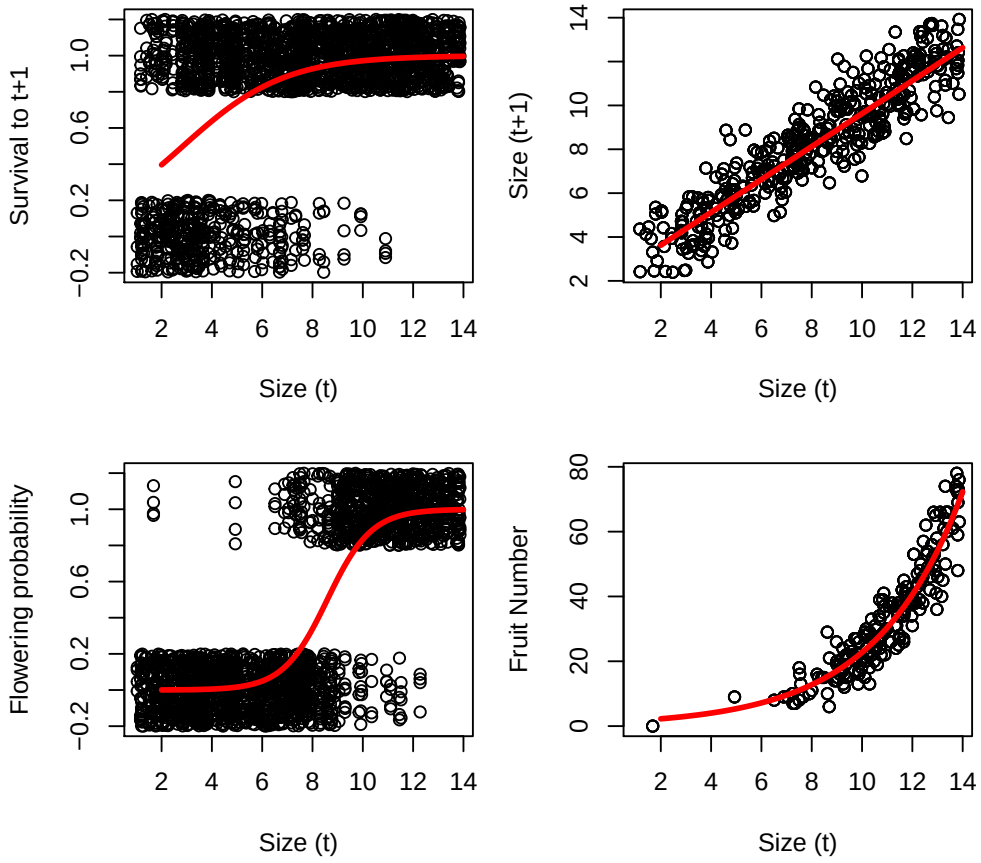

Figure 6.2: Regression lines for the size-dependent vital rates in the Actaea IPM.

Parameter values estimated from the original data are defined below. The use of additional parameters to describe the dependence of vital rates on soil potassium and seed predation is apparent from the vital rate functions (defined in section 6.4.2).

Model component	Function
Vital rates	
Survival ($s(z)$)	$\text{logit}(s) = -1.39 + 0.49\, z$
Growth ($g(z',z)$)	$z' = 2.13 + 0.71\, z + 0.013\, K_{conc}$, $\sigma^2 = 1.25$
Flowering (p_{flower})	$\text{logit}(p_{flower}) = -9.90 + 1.18\, z$
Fruit number ($f_{fruit}(z)$)	$\log(f_{fruit}) = 0.31 + 0.28\, z$
Seed number ($f_{seeds/fruit}(p_{pred})$)	$f_{seeds/fruit} = 0.39 p_{pred} + 9.26(1 - p_{pred})$
Germination (p_{estab})	$p_{estab} = 0.0062$
Seedling survival ($s_{seed\ surv}$)	$s_{seed\ surv} = 0.24$
Seedling size ($f_{recruit\ size}$)	$f_{recruit\ size}$: mean = 3.08, s.d. = 1.45
Covariates	
Potassium concentration (K_{conc})	$K_{conc} = 1.72\exp(3.01-(0.53*p_{spruce})$
Proportion spruce (p_{spruce})	$p_{spruce} = 1/(1+\exp(-0.09t+5))$
Proportion fruits predated (p_{pred})	$p_{pred} = 0.2$

Figure 6.3: Regression models used for A. spicata IPM.

After the vital rate models are built (see Appendix G for a discussion of regression model building techniques), the parameter values are included in the vector p.vec, for calculation of IPM kernels.

```
## a vector with the parameter values

p.vec<-rep(0,15)
p.vec[1:2]<- coef(surv.mod)[1:2]
names(p.vec)[1:2] <- c("surv.int", "surv.slope")
p.vec[3:5]<- coef(grow.mod)[1:3]
```

```
names(p.vec)[3:5] <- c("grow.int", "grow.size.slope", "grow.K.slope")
p.vec[6:7]<- coef(flow.mod)[1:2]
names(p.vec)[6:7] <- c("flow.int", "flow.slope")
p.vec[8:9]<- coef(fruit.mod)[1:2]
names(p.vec)[8:9] <- c("fr.int", "fr.slope")
p.vec[10]<- 3.0721075
names(p.vec)[10] <- "m.estab.size"
p.vec[11]<- 1.4488990
names(p.vec)[11] <- "sd.estab.size"
p.vec[12]<- 1.155625
names(p.vec)[12] <- "sd.growth"
p.vec[13]<- 0.0091550
names(p.vec)[13] <- "seedl.est"
p.vec[14]<- 0.2383389
names(p.vec)[14] <- "seedl.surv"
p.vec[15]<- 0.195905
names(p.vec)[15] <- "seed.pred"
```

6.4.2 Define functions to describe life history

We define the following functions that calculate vital rates (survival, growth, fertility) from the stored parameter values.

```
###### Probability of survival
sx <- function(x,params) {
   u<-exp(params["surv.int"] + params["surv.slope"]*x)
   return(u/(1+u))
 }
###### Growth function
gyx <- function(y,x,params) {
   dnorm(y, mean = params["grow.int"] + params["grow.size.slope"]*x + params["grow.K.slope"]*
 }
# Fertility function, expected number of seeds produced by a size x ind
fx<-function(x,params) {

   #probability of flowering:
   u<-exp(params["flow.int"]+params["flow.slope"]*x)

   #number of fruits per flowering ind:
   n.fruits<-exp(params["fr.int"]+params["fr.slope"]*x)
   n.fruits[n.fruits<0]<-0
```

```
   n.seeds.per.fruit<-params["seed.pred"]*0.38901+
     (1-params["seed.pred"])*9.61368
     # .38 & 9.61 are mean seed numbers in predated vs. unpredated fruits

   return(u/(1+u)*n.fruits*n.seeds.per.fruit)
 }
###### Probability of seedling survival and size distribution
py3<-function(y,params) {
   return(params["seedl.surv"]*
     dnorm(y,params["m.estab.size"],params["sd.estab.size"])/
       (1-pnorm(minsize,params["m.estab.size"],params["sd.estab.size"])))
 } # divided by the probability of being larger than the minimum
  # size to avoid eviction
```

6.4.3 Make a kernel

To construct an IPM kernel, we first define the boundary points (b; the limits of the size bins defining the kernel), mesh points (y; the centers of these bins, and the points at which the kernel is evaluated, under the midpoint rule of numerical integration), and step size (h; the widths of the bins). The integration limits (min.size and max.size) span the range of sizes observed in the data set, and then some (10 percent smaller/larger than the smallest/largest observed individual). Note that this is exactly the same procedure used in Appendix A.

```
# number of classes, or points for midpoint rule approximation
n.size = 100
# tolerance for iterations
tol = 1.e-8
# minimum and maximum sizes (0.9*min & 1.1*max size from data)
minsize = 0.9234374; maxsize = 15.27885
# calculate h - the width of the size classes in the matrix (discretized kernel)
L= minsize; U= maxsize; n <- n.size
b = L+c(0:n)*(U-L)/n
y = 0.5*(b[1:n]+b[2:(n+1)])
h = y[2]-y[1]
```

When the vital rate functions include covariates, as here, it is necessary to specify the values of the covariates. For example, if covariates describe the environment, a different kernel is built for each unique set of environmental conditions, which assumes that all individuals experience the same environment. The relationship between forest composition and population growth rate is modeled when constructing the IPM kernels, by modeling soil potassium as a function of spruce proportion (from Dahlgren and Ehrlen 2011). The following code contains a loop in which the kernel is rebuilt for different environments, i.e., different proportions of spruce in the tree community. The

code returns population growth rate and spruce proportion for 120 years, describing a successional sequence from 0 to 100% spruce. From this output, it is evident how average *Actaea* population growth rate is expected to change as a function of the proportion of spruce, and the corresponding number of years since colonization by spruce. Soil potassium (K.conc) is modeled as a function of time since spruce colonization (sim.yr) and spruce proportion as in Dahlgren and Ehrlen (2011) and the table in Fig. 6.3. In the code, y = z' and x = z.

```
# Loop to calculate lambda over forest succession from deciduous to spruce forest

# set spruce proportion and K.conc
spruce<-0
K.conc<-0
for (sim.yr in 1:120){
   spruce <- 1/(1+exp(-0.09*sim.yr+5))
   K.conc <- exp(3.00932-0.53267*spruce)*1.722324

   #--------------------
   # kernel components
   #--------------------

   G<-array(0,dim=c(n.size,n.size))
   ## runs gyx for each combination of possible sizes
   G<-h*outer(y,y,gyx,params=p.vec)

   # growth correction, rerouting growth to sizes outside the
   # allowed range to the extreme sizes (avoiding eviction):
   for(i in 1:(n.size/2)) G[1,i]<-G[1,i]+1-sum(G[,i])
   for(i in (n.size/2+1):n.size) G[n.size,i]<-G[n.size,i]+1-sum(G[,i])

   # survival vector
   S<-sx(y,p.vec)

   # fecundity vector
   F<-fx(y,p.vec)*h

   # survival-growth matrix
   P<-array(0,dim=c(2+n.size,2+n.size))
     # first 2 rows/columns are seed and seedling transitions
   P[2,1]<-p.vec["seedl.est"]
   P[3:(2+n.size),2]<-py3(y,p.vec)
```

```
    for(i in 3:(2+n.size))
      P[i,3:(2+n.size)]<-S*G[(i-2),]

    #-----------------------
    # Population iterations
    #-----------------------

    # iteration function
    iteration <- function(Nt) {
      Nt1=array(0,dim=c(2+n.size))
      Nt1[1]<-Nt1[1]+sum(F%*%Nt[3:(n.size+2)])        # fecundity
      Nt1<-Nt1+as.vector(P%*%Nt) # survival-growth
      return(Nt1)
    }

    # calculating lambda and stable size distribution using model iterations
    Nt=array(1,dim=c(2+n.size))
    Nt=Nt/sum(Nt)
    qmax=1000; lam=1
    while(qmax>tol) {
      Nt1=iteration(Nt) ## calling the iteration function
      qmax=sum(abs(Nt1-lam*Nt))
      lam=sum(Nt1)
      Nt=Nt1/lam
    }
    stable.dist=Nt/sum(Nt); lam.stable=lam

    # printing lambdas and spruce proportions for all years
    cat("Year:", sim.yr, "Proportion spruce:" , spruce, "Lambda:", lam, "\n")
  }
Year: 1 Proportion spruce: 0.007318533 Lambda: 1.014777
Year: 2 Proportion spruce: 0.008002235 Lambda: 1.014736
Year: 3 Proportion spruce: 0.008749246 Lambda: 1.014691
Year: 4 Proportion spruce: 0.009565319 Lambda: 1.014641
Year: 5 Proportion spruce: 0.01045671 Lambda: 1.014588
Year: 6 Proportion spruce: 0.0114302 Lambda: 1.014529
Year: 7 Proportion spruce: 0.01249319 Lambda: 1.014465
Year: 8 Proportion spruce: 0.01365366 Lambda: 1.014395
Year: 9 Proportion spruce: 0.0149203 Lambda: 1.014319
```

```
Year: 10 Proportion spruce: 0.0163025 Lambda: 1.014236
Year: 11 Proportion spruce: 0.01781043 Lambda: 1.014145
Year: 12 Proportion spruce: 0.01945508 Lambda: 1.014046
Year: 13 Proportion spruce: 0.02124832 Lambda: 1.013938
Year: 14 Proportion spruce: 0.02320294 Lambda: 1.013821
Year: 15 Proportion spruce: 0.0253327 Lambda: 1.013693
Year: 16 Proportion spruce: 0.02765242 Lambda: 1.013554
Year: 17 Proportion spruce: 0.03017798 Lambda: 1.013403
Year: 18 Proportion spruce: 0.03292639 Lambda: 1.013238
Year: 19 Proportion spruce: 0.03591585 Lambda: 1.013059
Year: 20 Proportion spruce: 0.03916572 Lambda: 1.012865
Year: 21 Proportion spruce: 0.04269664 Lambda: 1.012655
Year: 22 Proportion spruce: 0.04653047 Lambda: 1.012426
Year: 23 Proportion spruce: 0.05069032 Lambda: 1.012179
Year: 24 Proportion spruce: 0.05520054 Lambda: 1.011911
Year: 25 Proportion spruce: 0.06008665 Lambda: 1.011621
Year: 26 Proportion spruce: 0.06537533 Lambda: 1.011308
Year: 27 Proportion spruce: 0.0710943 Lambda: 1.01097
Year: 28 Proportion spruce: 0.0772722 Lambda: 1.010606
Year: 29 Proportion spruce: 0.08393843 Lambda: 1.010213
Year: 30 Proportion spruce: 0.09112296 Lambda: 1.009792
Year: 31 Proportion spruce: 0.09885607 Lambda: 1.009339
Year: 32 Proportion spruce: 0.1071681 Lambda: 1.008854
Year: 33 Proportion spruce: 0.1160889 Lambda: 1.008335
Year: 34 Proportion spruce: 0.1256479 Lambda: 1.007781
Year: 35 Proportion spruce: 0.1358729 Lambda: 1.00719
Year: 36 Proportion spruce: 0.1467903 Lambda: 1.006561
Year: 37 Proportion spruce: 0.1584242 Lambda: 1.005894
Year: 38 Proportion spruce: 0.1707955 Lambda: 1.005188
Year: 39 Proportion spruce: 0.1839217 Lambda: 1.004442
Year: 40 Proportion spruce: 0.1978161 Lambda: 1.003656
Year: 41 Proportion spruce: 0.2124868 Lambda: 1.002831
Year: 42 Proportion spruce: 0.2279365 Lambda: 1.001966
Year: 43 Proportion spruce: 0.2441611 Lambda: 1.001064
Year: 44 Proportion spruce: 0.26115 Lambda: 1.000124
Year: 45 Proportion spruce: 0.2788848 Lambda: 0.9991497
Year: 46 Proportion spruce: 0.2973393 Lambda: 0.9981424
Year: 47 Proportion spruce: 0.3164791 Lambda: 0.9971048
Year: 48 Proportion spruce: 0.3362613 Lambda: 0.9960402
```

```
Year: 49 Proportion spruce: 0.3566349 Lambda: 0.9949518
Year: 50 Proportion spruce: 0.3775407 Lambda: 0.9938435
Year: 51 Proportion spruce: 0.3989121 Lambda: 0.9927194
Year: 52 Proportion spruce: 0.4206757 Lambda: 0.991584
Year: 53 Proportion spruce: 0.4427521 Lambda: 0.9904416
Year: 54 Proportion spruce: 0.4650571 Lambda: 0.989297
Year: 55 Proportion spruce: 0.4875026 Lambda: 0.988155
Year: 56 Proportion spruce: 0.5099987 Lambda: 0.9870201
Year: 57 Proportion spruce: 0.5324543 Lambda: 0.9858969
Year: 58 Proportion spruce: 0.5547792 Lambda: 0.9847898
Year: 59 Proportion spruce: 0.5768853 Lambda: 0.9837028
Year: 60 Proportion spruce: 0.5986877 Lambda: 0.9826398
Year: 61 Proportion spruce: 0.6201064 Lambda: 0.9816043
Year: 62 Proportion spruce: 0.6410674 Lambda: 0.9805991
Year: 63 Proportion spruce: 0.6615032 Lambda: 0.9796271
Year: 64 Proportion spruce: 0.6813537 Lambda: 0.9786902
Year: 65 Proportion spruce: 0.7005671 Lambda: 0.9777904
Year: 66 Proportion spruce: 0.7190997 Lambda: 0.9769288
Year: 67 Proportion spruce: 0.7369159 Lambda: 0.9761065
Year: 68 Proportion spruce: 0.7539887 Lambda: 0.9753239
Year: 69 Proportion spruce: 0.7702989 Lambda: 0.9745812
Year: 70 Proportion spruce: 0.785835 Lambda: 0.9738783
Year: 71 Proportion spruce: 0.8005922 Lambda: 0.9732145
Year: 72 Proportion spruce: 0.8145726 Lambda: 0.9725894
Year: 73 Proportion spruce: 0.8277836 Lambda: 0.9720018
Year: 74 Proportion spruce: 0.840238 Lambda: 0.9714507
Year: 75 Proportion spruce: 0.8519528 Lambda: 0.9709349
Year: 76 Proportion spruce: 0.8629487 Lambda: 0.970453
Year: 77 Proportion spruce: 0.8732494 Lambda: 0.9700034
Year: 78 Proportion spruce: 0.882881 Lambda: 0.9695847
Year: 79 Proportion spruce: 0.8918713 Lambda: 0.9691953
Year: 80 Proportion spruce: 0.9002495 Lambda: 0.9688338
Year: 81 Proportion spruce: 0.9080455 Lambda: 0.9684985
Year: 82 Proportion spruce: 0.9152894 Lambda: 0.9681879
Year: 83 Proportion spruce: 0.9220118 Lambda: 0.9679004
Year: 84 Proportion spruce: 0.9282425 Lambda: 0.9676347
Year: 85 Proportion spruce: 0.934011 Lambda: 0.9673893
Year: 86 Proportion spruce: 0.9393461 Lambda: 0.9671628
Year: 87 Proportion spruce: 0.9442756 Lambda: 0.966954
```

```
Year: 88 Proportion spruce: 0.9488263 Lambda: 0.9667616
Year: 89 Proportion spruce: 0.9530239 Lambda: 0.9665845
Year: 90 Proportion spruce: 0.9568927 Lambda: 0.9664215
Year: 91 Proportion spruce: 0.9604562 Lambda: 0.9662716
Year: 92 Proportion spruce: 0.9637363 Lambda: 0.9661338
Year: 93 Proportion spruce: 0.9667537 Lambda: 0.9660072
Year: 94 Proportion spruce: 0.969528 Lambda: 0.965891
Year: 95 Proportion spruce: 0.9720774 Lambda: 0.9657842
Year: 96 Proportion spruce: 0.9744192 Lambda: 0.9656863
Year: 97 Proportion spruce: 0.9765693 Lambda: 0.9655965
Year: 98 Proportion spruce: 0.9785427 Lambda: 0.9655141
Year: 99 Proportion spruce: 0.9803532 Lambda: 0.9654386
Year: 100 Proportion spruce: 0.9820138 Lambda: 0.9653694
Year: 101 Proportion spruce: 0.9835364 Lambda: 0.9653059
Year: 102 Proportion spruce: 0.984932 Lambda: 0.9652478
Year: 103 Proportion spruce: 0.986211 Lambda: 0.9651946
Year: 104 Proportion spruce: 0.9873828 Lambda: 0.9651459
Year: 105 Proportion spruce: 0.9884562 Lambda: 0.9651013
Year: 106 Proportion spruce: 0.9894393 Lambda: 0.9650604
Year: 107 Proportion spruce: 0.9903395 Lambda: 0.9650231
Year: 108 Proportion spruce: 0.9911636 Lambda: 0.9649888
Year: 109 Proportion spruce: 0.991918 Lambda: 0.9649575
Year: 110 Proportion spruce: 0.9926085 Lambda: 0.9649289
Year: 111 Proportion spruce: 0.9932403 Lambda: 0.9649027
Year: 112 Proportion spruce: 0.9938185 Lambda: 0.9648787
Year: 113 Proportion spruce: 0.9943476 Lambda: 0.9648568
Year: 114 Proportion spruce: 0.9948315 Lambda: 0.9648367
Year: 115 Proportion spruce: 0.9952743 Lambda: 0.9648183
Year: 116 Proportion spruce: 0.9956793 Lambda: 0.9648016
Year: 117 Proportion spruce: 0.9960497 Lambda: 0.9647862
Year: 118 Proportion spruce: 0.9963884 Lambda: 0.9647722
Year: 119 Proportion spruce: 0.9966983 Lambda: 0.9647594
Year: 120 Proportion spruce: 0.9969816 Lambda: 0.9647476
```

Assuming that forest succession follows the trajectory published by Dahlgren and Ehrlen (2011), we can predict the distribution of *Actaea* population growth rate in the future. Fig. 6.4 illustrates how a relatively small change in a soil nutrient affecting individual growth can influence a species' potential geographic distribution. Panel a in Fig. 6.4 shows the population growth rate expected for *Actaea*, given a preliminary survey of current forest composition on the Naset peninsula, combined with the above output from the soil potassium-dependent IPM. By inferring the number of years

since spruce colonization from the map of current forest composition, the spatial distribution of population growth rate can be simulated (forecast) 25 years into the future (panel b, Fig. 6.4). The projected decline in the distribution of *A. spicata* (panel a vs. b, Fig. 6.4) arises from the sensitivity of λ to individual growth. This illustrates one consequence of forest succession, which has been altered by 20th (and 21st) century patterns of land use. A reduction in anthropogenic disturbance means that deciduous-dominated forests are no longer generated as they were in the past.

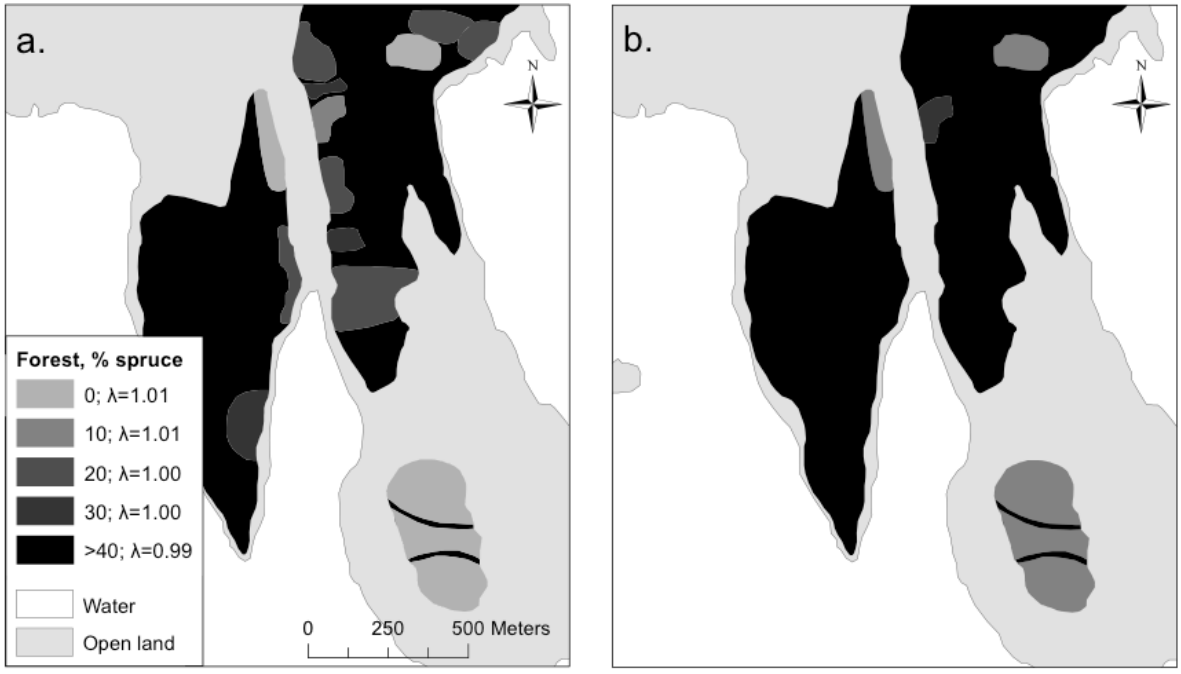

Figure 6.4: Predicted distribution of the demographic niche of Actaea spicata on the Naset peninsula in Tullgarn natural reserve in (a) 2013 and (b) 2038, based on the effect of forest composition on soil potassium concentration and expected change in forest composition. Soil potassium affects population growth rate (λ) according to the potassium-dependent IPM described in this appendix. Soil potassium concentration was modeled as a function of the proportion of spruce in the forest, which increases over time (Fig. 6.3; Dahlgren and Ehrlen 2011). At 0% spruce, lambda = 1.01, whereas at approximately 40% spruce, lambda = 0.99. Since the bootstrapped 90% confidence interval for lambda is +/- 0.01, spruce composition >40% is considered unsuitable habitat for A. spicata.

6.4.4 Improving the model

Though not immediately available, additional data are one way to improve analyses such as this one. Here parameters were estimated from two annual transitions (three censuses); with more years of data, extreme years with large effects on population dynamics are more likely to be sampled. Additional data would also form the basis for developing a stochastic model with different yearly kernels (see Appendix D). Density dependence, too, may be important to account for (see Ellner and Rees 2006 for an example on how to include this in IPMs). For example, if density declines

and negative interspecific density dependence is strong, the population growth rate will be higher than inferred from a density-independent model (Dahlgren and Ehrlen 2011). However, we tested for density-dependence in this system and were unable to detect an effect. With the available data, the IPM presented here may also be improved by using regression splines to relax the linearity assumptions of GLMs (Dahlgren et al. 2011). Accounting for relationships between vital rates and state variable that are non-monotonic has been shown to have potentially large effects on population growth rates (and other model predictions such as sensitivities).

6.5 Conclusions

In this case study, we have demonstrated how IPMs can be based on regression models that include covariates, thereby making the IPM kernel depend on those covariates. This illustrates the potential of IPMs to capture mechanistic relationships between the environment and population dynamics through effects on vital rates of individuals.

6.6 Acknowledgements

The code for the IPM was based on Ellner and Rees (2006) and Dahlgren and Ehrlen (2009, 2011).

6.7 References

- Dahlgren, J.P. & Ehrlen, J. (2011). Incorporating environmental change over succession in an integral projection model of population dynamics of a forest herb. Oikos, 120, 1183-1190.
- Dahlgren, J.P. & Ehrlen, J. (2009). Linking environmental variation to population dynamics of a forest herb. Journal of Ecology, 97, 666-674.
- Ellner, S. & Rees, M. (2006). Integral projection models for species with complex demography. American Naturalist, 167, 410-428.
- Williams, J., Auge, H. & Maron, J. (2010). Testing hypotheses for exotic plant success: parallel experiments in the native and introduced ranges. Ecology, 91, 1355-1366.

Chapter 7

Appendix G: Evolutionary demography with monocarpic perennials

C.J.E. Metcalf (charlotte.metcalf@zoo.ox.ac.uk)

Abstract

Integral Projection Models have been widely used for the analysis of Evolutionarily Stable Strategies. In this Appendix we introduce some of the associated methods and underlying logic.

7.1 Introduction

Evolutionarily Stable Strategy (ESS) analysis illustrates how complex inference can emerge from manipulating vital rate functions to characterize demographic tradeoffs (Childs et al. 2003, Rees et al. 2006, Childs et al. 2011). This approach examines the evolution of life histories by comparing the expected fitness of different life history strategies, typically encoded in the parameters of the vital rate regressions. The aim is to determine the strategy that, once established, cannot be invaded by any other strategy, the so-called ESS. The key element for such an analysis is the definition of a trade-off associated with the focal trait. In the absence of tradeoffs, we can only predict that the trait should evolve to its largest or smallest possible value. For example, if no cost of reproduction is defined, the optimal size at first reproduction is the smallest possible size, to allow individuals to start reproducing as early as possible. Conversely, if no cost of offspring size is defined, the optimal offspring size will be the largest possible size. A variety of tradeoffs might exist: between growth and reproduction, between offspring number and offspring size, or as in this example, between current and future reproductive potential in monocarpic herbs. These models assume that some degree of heritability underlies the trait of interest, otherwise evolution cannot be expected to act.

Monocarpic plants are plant species where reproduction is fatal. The timing of reproduction is therefore under strong selection. It is also subject to a very simple trade-off. For a monocarpic, perennial plant, the trade-off necessary for evolutionary analyses of size at flowering is relatively straightforward: the longer individuals wait before flowering, the larger they grow, and the more seeds they produce. However, if they wait too long, they may die without ever having reproduced (Metcalf et al. 2003). Predicting the optimal size at which individuals should flower requires estimating the demographic consequences of flowering across a range of sizes. This requires the process-based demography employed by IPMs (Rees and Rose 2002).

7.2 Study System/Objective

Here, we use two years of field data collected on monocarpic species including *C. arvense* at Silwood park to demonstrate how the optimal flowering size can be identified (i.e.,which flowering size leads to highest fitness), and further, which flowering size is expected to be evolutionarily stable (i.e., cannot be invaded by any other strategy).

7.3 Data

The data was collected at Silwood Park, Imperial College, UK, on a range of monocarpic species (*Senecio jacobea*, *Cirsium palustrae*, *Cirsium arvense*, *Cirsium vulgare*, *Digitalis purpurea*, *Verbascum blattaria*, *Verbascum thapsus*, *Arctium minus*). Metcalf et al. 2006 describe this in detail:

'Eight sites were chosen, encompassing a broad range of habitat types, to maximize the chance of detecting variation in allocation strategies. In total eight species were investigated: Arctium minus

Bernh.; Cirsium arvense (L.) Scop.; Cirsium palustre (L.) Scop.; Cirsium vulgare (Savi.) Ten.; Digitalis purpurea L.; Senecio jacobaea L.; Verbascum blattaria L.; Verbascum thapsus L. In May 2003 we marked a broad size-range of individuals from all species present at each site (a total of 1100 individuals). For each individual we measured root crown diameter, length of the longest leaf, and maximum rosette diameter. We also recorded whether plants had flowered or not between May and August 2003. In May 2004 we recorded which individuals had died (excluding those that had flowered), and measured surviving individuals. This is referred to as the between-year data set.

Many of the marked plants flowered in the first year, and relatively few individuals died, making it difficult to estimate species-specific survival schedules. We therefore marked an additional 1300 non-flowering individuals of five species (C. arvense, C. vulgare, D. purpurea, S. jacobaea, V. blattaria) at three sites in June 2004, and re-measured them in late August 2004 to explore growth and survival. We also recorded flowering, and counted the number of flowering heads for 30 individuals of each species to explore reproductive allometries. This is referred to as the within-year data set.

The life history of S. jacobaea may range from a strict monocarp to an iteroparous perennial (Gillman & Crawley 1990), and this is also the case for D. purpurea (van Baalen & Prins 1983). All flowering individuals were, however, excluded from the within-year data set, so reproduction did not affect the allocation processes. Cirsium arvense reproduces vegetatively via lateral rhizomatous roots (Kluth, Kruess & Tscharntke 2003), and between-clone allocation may obscure the growthâĂŞ-survival allocation patterns of interest. However, the identity of genetic individuals could not be determined without disturbing the study population, so we assumed this effect was negligible.'

7.4 Analysis

The analyses are organized into the following sections:

1. Setting up the data
2. Identify appropriate statistical models for the key demographic functions
3. Building an IPM for monocarpic species
4. Setting the population at equilibrium
5. Exploring the IPM
6. Identifying the optimal flowering size
7. Adaptive dynamics
8. Fitness in a stochastic environment

7.4.1 Setting up the data

The following function loads the package IPMpack, and then the data, and reveals the header line:

```
library(IPMpack)
data(dataIPMpackSilwood)
head(data(dataIPMpackSilwood))
[1] "dataIPMpackSilwood"
```

which reveals 3 possible size measures: rootcrown diameter `rtcr`, length of the longest leaf `ll` and diameter `rosetteDiam`. Some IPMpack functions require data with specific column names; in particular, a column called `size` and a column called `sizeNext`. The next lines create a data-frame `dff`, containing columns with these names, that hold the desired size measure, here chosen to be log(root crown).

```
dff <- dataIPMpackSilwood
dff$size <- log(dff$rtcr)
dff$sizeNext <- log(dff$rtcrNext)
```

Finally, because these are monocarps, and we are interested in the flowering strategy, it does not make sense to equate "mortality" and "flowering" (mortality on flowering is a positive outcome, because you have successfully reproduced; but mortality without flowering is a negative one). However, at the moment, in the data where `flowered` is equal to 1, `surv` is equal to zero. This can be altered via:

```
dff$surv[dff$flowered==1] <- 1
```

Here, we store this data-set, and create a data-frame with a subset of information for *Cirsium arvense* for more detailed analysis.

```
dff.store <- dff
dff <- dff[dff$Species=="C.arvense",]
```

7.4.2 Identify appropriate statistical models for the key demographic functions

IPMpack uses growth, survival and fertility objects to build IPMs. See Example 2 in Section II.A. of the main text for a description of the equations used below. Growth and survival objects can be constructed using:

```
gr1 <- makeGrowthObj(dataf = dff,
                     Formula=sizeNext~size,
                     regType="constantVar",
                     Family="gaussian")
sv1 <- makeSurvObj(dff, Formula = surv~size)
```

This assumes that the covariate `size` is the appropriate explanatory variable for both growth (modeled via size at the next census interval with a linear regression) and survival (modeled with a logistic regression). Model comparison can be implemented via 7.1:

Lower AIC values indicate a better fit; confirming that `size` alone provides an appropriate model, as implemented to create `gr1` and `sv1`.

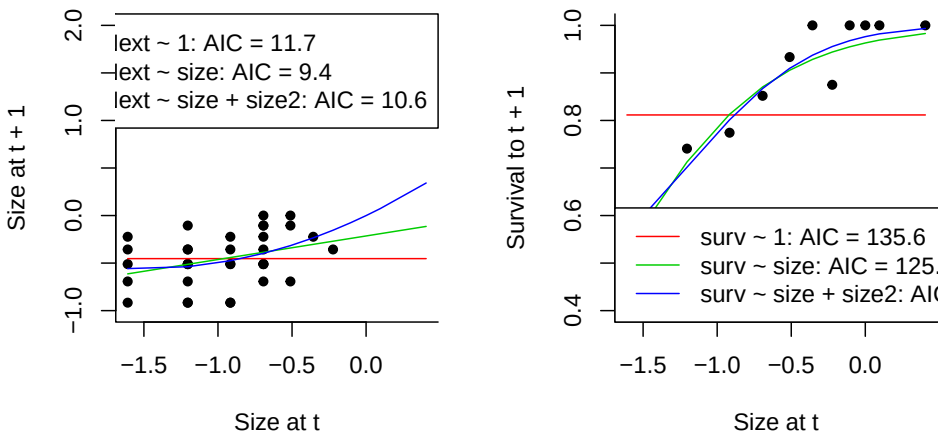

Figure 7.1: Model comparison

The data-set contains no information on seed output. However, for many plants, seed output scales approximately with biomass (Klinkhamer et al. 1992). As the size measure is taken on a log scale (see above), and the size measure is a diameter (to which biomass will square to approximately the power 2), the slope of the line predicting log seed production should be approximately 2. The choice of intercept can be arbitrary initially. As discussed below, this will not affect predictions of the evolutionarily stable flowering strategy, under some assumptions about the operation of density dependence.

Not all individuals produce seeds all the time, so for every size, we also need to multiply the predicted number of seeds produced by the probability that an individual of that size flowers. The column `flowered`, is zero if an individual did not flower in that year, and 1 otherwise. We can fit this using a logistic regression:

```
fit.flower <- glm(flowered~size, data=dff, family=binomial)
```

We can now create a fecundity object, specifying these two fertility relationships: seed production, and flowering probability. Seed production will have a slope of 2 and an arbitrarily chosen intercept, here set to 5 (equivalent to defining number of seeds produced by individuals of size x as $fec1 = exp(5+2x)$, keeping in mind that x is log transform of the size measure, as set above). This is specified in the first element of the list provided for the argument `coeff` in the `makeFecObj` below. The response variable for the seed production function ($fec1$) will be subjected to a log transform, specified via the first element in the list of the argument `Transform`. Note that it is not necessary to specify the variance in the seed production function (or indeed any other fitted function in fertility objects), since only the mean is used for the fertility component in an IPM. Flowering probability ($fec2$) takes the coefficients obtained in `fit.flower`, above, and will also

be fitted with binomial errors, specified via `Family`, with these various elements appearing as the second element in lists and vectors below:

```
fv1 <- makeFecObj(Formula=list(fec1~size,fec2~size),
                  coeff=list(c(5,2),
                  c(fit.flower$coeff[1],fit.flower$coeff[2])),
                  Family = c("gaussian","binomial"),
                  Transform = c("log","none"),
                  meanOffspringSize = as.numeric(quantile(dff$size,0.25, na.rm=TRUE)),
                  sdOffspringSize = 0.5)
```

To build a fertility object also requires information on the distribution of offspring sizes, and here, in the absence of information in the data set, we are specifying that the mean size of seedling recruits is defined as the quartile of the observed size distribution (in the argument `meanOffspringSize`), and we have chosen an arbitrary standard deviation of offspring size of 0.5 (in the argument `sdOffspringSize`).

Finally, the default in IPMpack is to assume that 100% of offspring are produced into the "continuous" stage (there is always a "continuous" stage in IPMpack in an IPM); the relevant help files in IPMpack can be consulted for implementing more complex life histories.

7.4.3 Building an IPM for monocarpic species

A demographic model describing growth and survival transitions from these objects can be specified via:

```
Pmatrix <- makeIPMPmatrix(nBigMatrix = 100,
                          minSize = -2,
                          maxSize = 2,growObj = gr1,
                          survObj = sv1,
                          correction="constant")
```

where `nBigMatrix` is the number of bins used (and output should be carefully checked to test that this is sufficient), `minSize` and `maxSize` define the limits of the IPM (which should usually extend to beyond the smallest and largest size measurements; since we are operating on a log scale, `minSize` may be negative) and the growth and survival objects are as above. The function `diagnosticsPmatrix` provides a series of plots indicative of whether bin choice and size range is adequate.

Note that while this will work for most species, there is a missing component for monocarps. Another way in which individuals leave the population is via flowering, since flowering is fatal. Each column of the IPM thus needs to be multiplied by one minus the probability of flowering as predicted for the size reflecting corresponding bin.

```
prob.not.flower <- 1-predict(fit.flower,
                             newdata=data.frame(size=Pmatrix@meshpoints),
                             type="response")
```

```
Pmatrix <- t(t(Pmatrix)*prob.not.flower)
```

The transposes above exploit the way R implements matrix x vector operations. We can also create a transition matrix describing movement between sizes attributable to fertility:

```
Fmatrix <- makeIPMFmatrix(nBigMatrix = 100,
                          minSize = -2, maxSize = 2,
                          fecObj = fv1,
                          correction="constant")
```

Note that we are assuming pre-census fertility (see help for `makeIPMFmatrix` for alternatives). By adding survival-growth and fertility transitions, we can now define a full Integral Projection Model:

```
IPM <- Pmatrix + Fmatrix
```

7.4.4 Setting the population at equilibrium

The least well-informed part of our model is the fertility component. It is reasonable to assume that the population is approximately at equilibrium. In many monocarpic species, it has been suggested that density dependence acts predominantly at the seed establishment stage (Metcalf et al. 2003; but see Sletvold, 2005). To set the population at equilibrium with this condition, we can find the value of the probability of seed establishment that sets the R_0 to 1, where R_0 is the net reproductive number, or the number of offspring produced on average by an individual over its lifespan. To do this, we define:

```
p.est <- (1/R0Calc(Pmatrix,Fmatrix))
p.est
```

```
[1] 0.02488588
```

We can then introduce `p.est` into the fertility object as a constant by which fertility is multiplied:

```
fv1@fecConstants <- data.frame(pe=p.est)
```

We need to update `vitalRatesPerOffspringType` to reflect this new element of fertility (specifying that all offspring are multiplied by 1, and then by "pe"):

```
fv1@vitalRatesPerOffspringType <- data.frame(continuous=rep(1,3))
rownames(fv1@vitalRatesPerOffspringType) <- c(fv1@fecNames,"pe")
```

The code below rebuilds the Fmatrix, and checks that the population is indeed at equilibrium (i.e., $R_0 = 1$)

```
Fmatrix <- makeIPMFmatrix(nBigMatrix = 100,
                          minSize = -2, maxSize = 2,
                          fecObj = fv1,
                          correction="constant")
R0Calc(Pmatrix,Fmatrix)
```

```
[1] 1
```

This assumption makes some of the uncertainty in the data less relevant, particularly as regards the intercept of the seed production function, which is essentially confounded with the probability of seed establishment. We redefine the IPM:

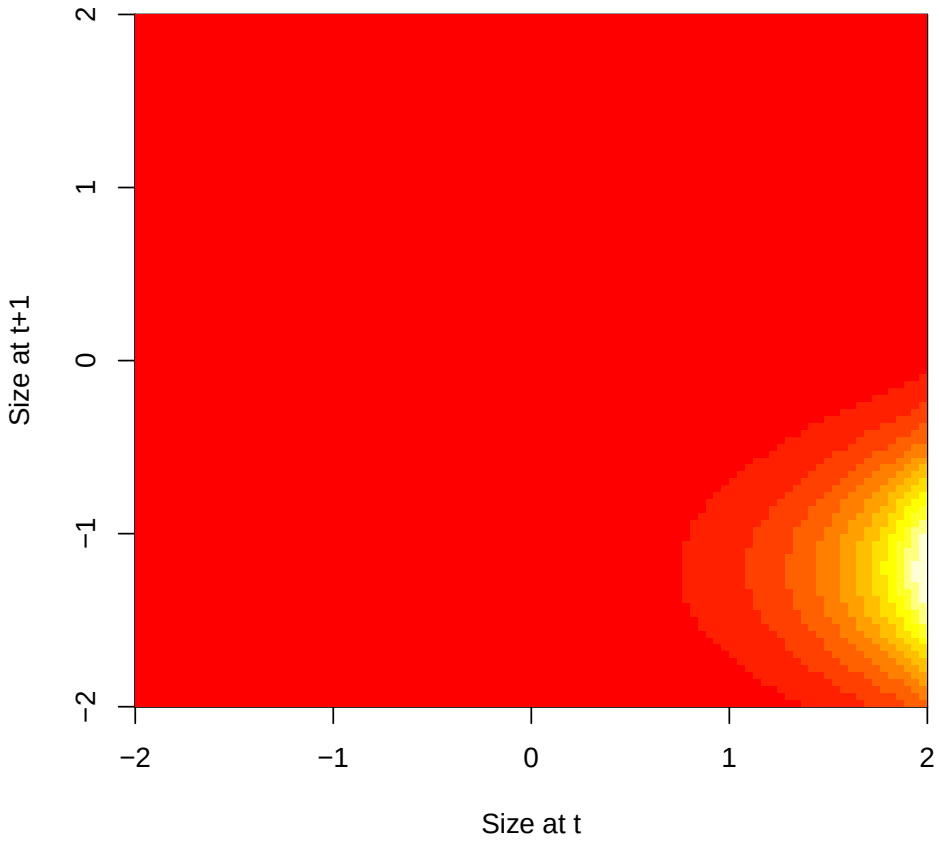

Figure 7.2: Integral Projection Model (completely dominated by fertility for this model, so that the growth diagonal is not visible)

```
IPM <- Pmatrix + Fmatrix
```

and plot the IPM matrix using `image` or similar (Figure 7.2).

```
image(Pmatrix@meshpoints, Pmatrix@meshpoints,
      t(IPM),
      xlab = "Size at t", ylab = "Size at t+1")
```

7.4.5 Exploring the IPM

Classic matrix population model results can be used to extract key aspects of the IPM, i.e., the growth rate λ the stable size distribution, sensitivity, elasticity, etc (Caswell, 2001).

```
lam <- eigen(IPM)$value[1]; print(lam)
```

```
[1] 1+0i
```

```
stabSize <- abs(eigen(IPM)$vector[ ,1])
sensitivity <- sens(IPM)
elasticity <- elas(IPM)
```

Note that the population growth rate is not informative here, as we have set the population at equilibrium. Outputs can be plotted against the meshpoints (Figure 7.3 p. 171).

```
par(mfrow = c(2, 2), bty = "l", pty = "s")
plot(Pmatrix@meshpoints, abs(eigen(IPM)$vector[ ,1]),
      type = "l", xlab = "Size",
      ylab = "Stable size distribution",
      xlim = range(dff$size, na.rm = T))
hist(dff$size,xlab="Size structure", col="grey",main="",
      ylab="Observed distribution")
image(Pmatrix@meshpoints, Pmatrix@meshpoints, sensitivity,
       xlab = "Size at t", ylab = "Size at t+1",
       main = "Sensitivity")
image(Pmatrix@meshpoints, Pmatrix@meshpoints, elasticity,
       xlab = "Size at t", ylab = "Size at t+1",
       main = "Elasticity")
```

here showing the predicted stable size distribution and the observed size distribution in the population side by side in the top row for comparison; elasticity and sensitivity on the bottom rows.

7.4.6 Identifying the optimal flowering size

The assumption that density dependence operates on seedling establishment is convenient for Evolutionary Stable Strategy (ESS) analysis since Mylius and Diekmann (1995) indicated that if density dependence operates on offspring establishment, the strategy that maximizes R_0 is the ESS (and not just the optimum in terms of a fitness measure - these two can be different).

Identifying the optimal flowering size therefore requires exploring a range of different flowering functions and calculation the corresponding R_0. We can do this by looping over a suite of different intercepts of the flowering function (bigger intercepts will lead to flowering at smaller sizes, see Figure 7.4)), and then rebuilding the T and F matrix (the T matrix will also change, because flowering leads to mortality) and then calculate the corresponding R_0. For convenience, firsts we store some of the elements that won't change over the loop:

```
Pmatrix.base <- makeIPMPmatrix(nBigMatrix = 100,
                               minSize = -2,
                               maxSize = 2,growObj = gr1,
                               survObj = sv1, correction="constant")
current.fit.flower <- fit.flower
fv1.base <- fv1
```

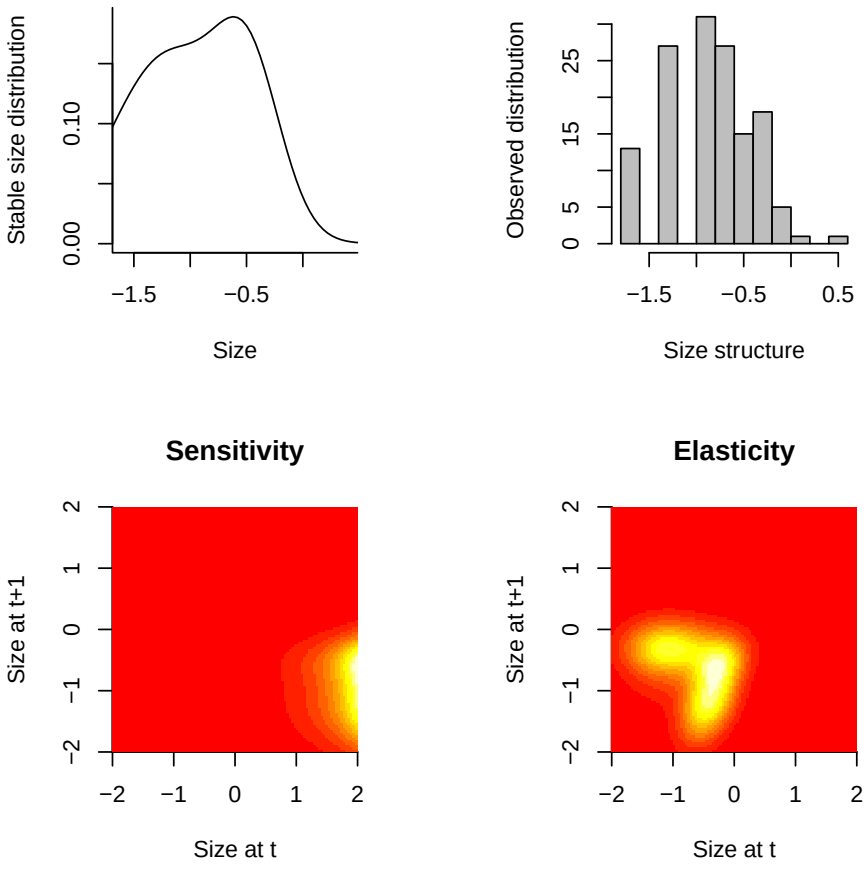

Figure 7.3: Measures off a full IPM: the stable size distribution, and observed size distribution for comparison; followed by the sensitivity and elasticity of the IPM.

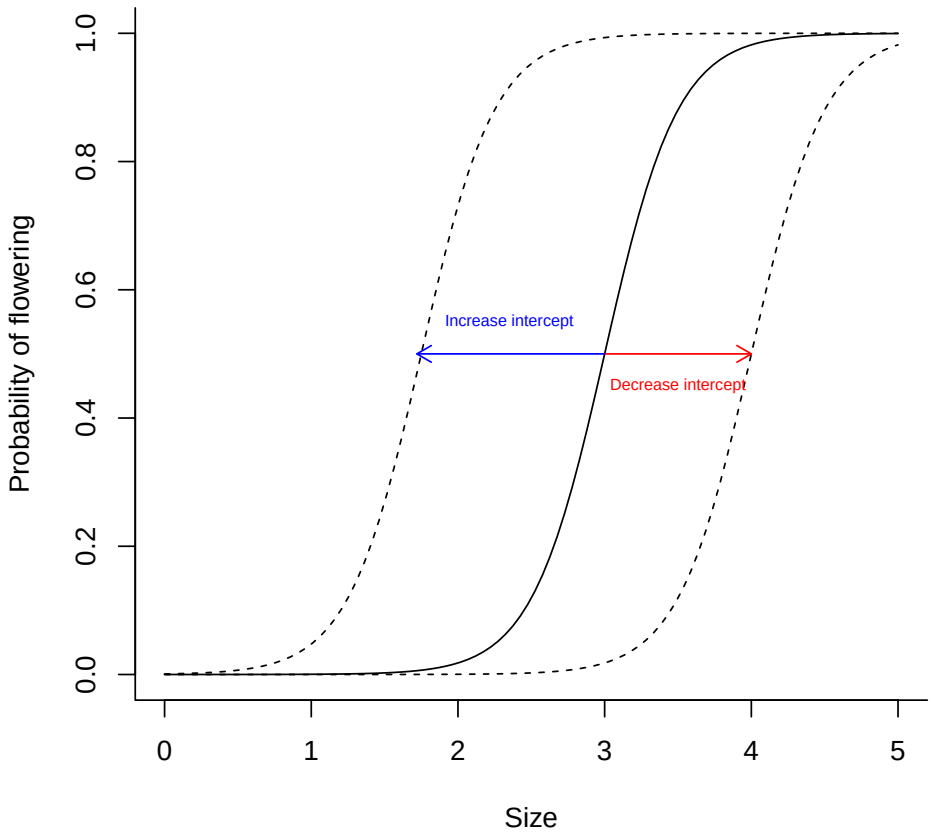

Figure 7.4: Illustration of the effect of increasing or decreasing the intercept of flowering function on the probability of flowering, and thereby size at flowering.

Then we create vectors and matrices to hold results of the loop and define the range of intercepts we want to explore:

```
store.R0 <- rep(NA,100)
test.intercept <- seq(-5,4,length=100)
store.flow <- matrix(NA,100,length(Pmatrix@meshpoints))
```

and finally we run the loop

```
for (k in 1:length(store.R0)) {
     #over-write intercept
     fit.flower$coefficients[1] <- test.intercept[k]
     fv1 <- fv1.base
     fv1@fitFec[[2]] <- fit.flower

     #rebuild Pmatrix
     prob.not.flower <- 1-predict(fit.flower,
                                  newdata=data.frame(size=Pmatrix@meshpoints),
                                  type="response")
     Pmatrix1 <- t(t(Pmatrix.base)*prob.not.flower)
     store.flow[k,] <- 1-prob.not.flower

     #rebuild Fmatrix
     Fmatrix1 <- makeIPMFmatrix(nBigMatrix = 100,
                                  minSize = -2, maxSize = 2,
                                  fecObj = fv1,
                                  correction="constant")
     #store R0
     store.R0[k] <- R0Calc(Pmatrix=Pmatrix1,Fmatrix=Fmatrix1)
 }
```

which can be plotted via:

```
par(mfrow=c(1,1),bty="l")
plot(test.intercept,store.R0,xlab="intercept tested",
      ylab=expression(R[0]), type="l")
best.intercept <- test.intercept[store.R0==max(store.R0)]
abline(v=best.intercept,lty=3)
abline(v=current.fit.flower$coefficients[1],lty=2,col=2)
abline(h=1,col="grey")
```

On this figure, the observed intercept for the relationship between size and probability of flowering for *Cirsium arvense* at Silwood is shown as a red vertical line, and the predicted ESS as a black vertical line. Because the predicted ESS intercept is smaller than the former (making the probability of flowering at an identical size smaller at the ESS than at the observed) this indicates

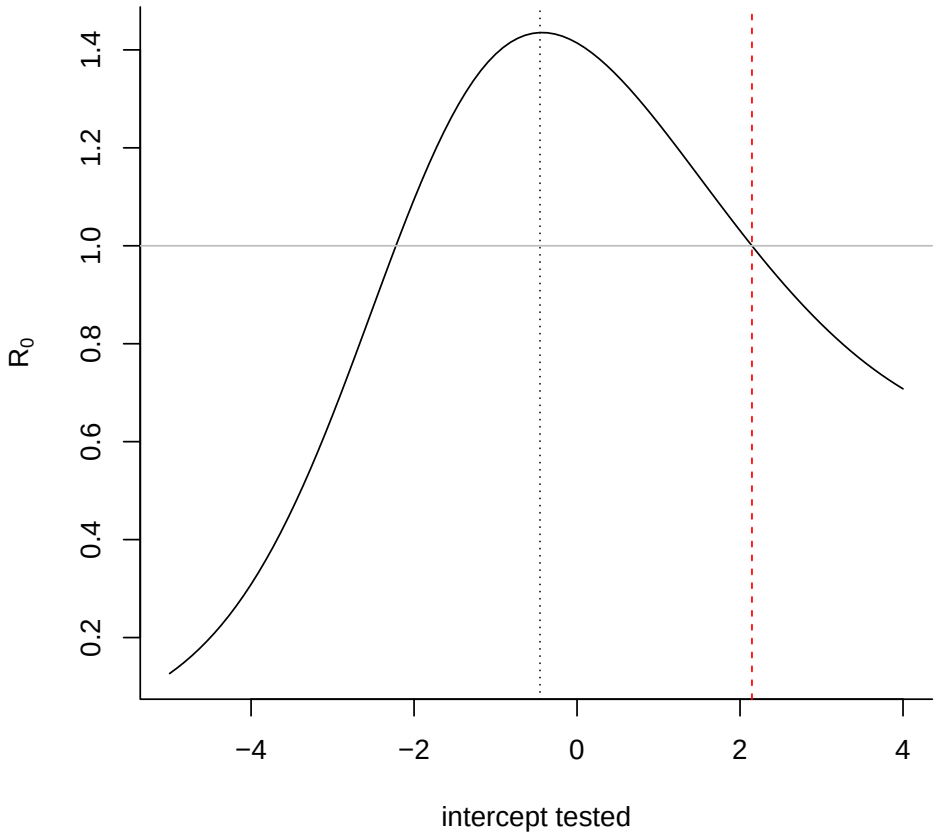

Figure 7.5: Fitness landscape

that the ES flowering strategy is actually larger than the observed flowering strategy, and we would expect plants to be delaying for longer than they actually are. Due to limitations in the data, this analysis necessarily ignores all complications of stochastic environments (Childs et al. 2004), seed predators (Metcalf et al. 2008; Rose et al. 2005), etc, which may partly account for this failure. See Metcalf et al. 2003 for a review.

In the simple case of constant environment with density dependence in seedling establishment, both the intercept of the relationship between size and log number of seeds (here set to 5, see above), and the probability of seed establishment (scaled so that the resident strategy is at equilibrium, see above) have the effect of moving the entire R_0 curve in Figure 7.5 up or down, but they do not change the shape of this curve - i.e., the location of the peak of this curve does not change. This occurs because, inherently, these two components do not affect the relationship between size and reproductive output, they just change the total output by a constant amount, and therefore they cannot alter the optimal flowering size.

7.4.7 Adaptive dynamics

Since density dependence is thought to be operating in these systems, which means that the success of any strategy depends on what other individuals in the population are doing, the outcome of evolutionary dynamics can also be studied using an approach known as "adaptive dynamics". Given the simple form of the density dependence considered, and since we focus on an analysis with constant environment, the outcome is in fact rather trivial (and identical to the ESS answer we found above), but for illustration, the methods are outlined developed here.

Adaptive dynamics is essentially a graphical approach to exploring the outcome of evolutionary dynamics in the situation where we assume that mutations are small, and ecology occurs on a much faster time-scale than evolution (so that we can separate the two time-scales).

Practically, the analysis consists of setting up a matrix that will contain, for all possible pairs of (discretized) resident and invader strategies, the population rate of increase of the invading strategy. The range of strategies will encompass the smallest to the largest flowering size of interest (i.e., intercepts of the flowering function ranging from -5 to 4 as above). The success of an invader will be established by determining its λ in a context where the probability of seedling establishment is set by the resident strategy. This follows because we assume that the invader is so rare on its first appearance that the density dependent environment is entirely determined by the resident.

The code below sets up a matrix to store the invader's λ and define the range of intercepts explored; and then runs the double loop, looping over all possible resident and invader strategies (this can be a bit slow):

```
store.lam <- matrix(NA,60,60)
test.intercept <- seq(-5,4,length=60)
for (j in 1:length(test.intercept)) {
     #set the resident strategy as the jth intercept and
     fit.flower$coefficients[1] <- test.intercept[j]
```

```
fv1 <- fv1.base
fv1@fitFec[[2]] <- fit.flower

#rebuild Pmatrix
prob.not.flower <- 1-predict(fit.flower,
                             newdata=data.frame(size=Pmatrix@meshpoints),
                             type="response")
Pmatrix1 <- t(t(Pmatrix.base)*prob.not.flower)

#rebuild Fmatrix
Fmatrix1 <- makeIPMFmatrix(nBigMatrix = 100,
                             minSize = -2, maxSize = 2,
                             fecObj = fv1,
                             correction="constant")

#calculate the p.est that would set the resident at equilibrium
p.est <- (1/R0Calc(Pmatrix1,Fmatrix1))

#loop over the invaders
for (k in 1:length(test.intercept)) {
    #over-write intercept with the kth intercept to set the invader
    fit.flower$coefficients[1] <- test.intercept[k]
    fv1 <- fv1.base
    fv1@fitFec[[2]] <- fit.flower

    #rebuild Pmatrix
    prob.not.flower <- 1-predict(fit.flower,
                                 newdata=data.frame(size=Pmatrix@meshpoints),
                                 type="response")
    Pmatrix1 <- t(t(Pmatrix.base)*prob.not.flower)

    #rebuild Fmatrix
    Fmatrix1 <- makeIPMFmatrix(nBigMatrix = 100,
                                 minSize = -2, maxSize = 2,
                                 fecObj = fv1,
                                 correction="constant")
    #store lambda that would result from
    # p.est imposed by resident (putting it in here
    # is a short-cut rather than adding it as a constant);
```

```
        # since multiplying up all the constants can be confusing....
        store.lam[j,k] <- Re(eigen(Pmatrix1+p.est*Fmatrix1)$value[1])
    }
}
```

To identify the ESS, we plot the areas where the invading strategy can successfully invade (corresponding to where $\lambda > 1$ for the invading strategy).

```
image(test.intercept,test.intercept,1*(store.lam>=1),
      xlab="Intercept of resident flowering strategy",
      ylab="Intercept of invader flowering strategy",
      col=grey(c(0.5,1)))
abline(v=best.intercept,lty=1,lwd=2,col="red")
abline(h=best.intercept,lty=1,lwd=2,col="red")
```

The first conclusion that can be drawn from this is that there is a convergent stable evolutionary strategy. This emerges because there is a strategy that cannot be invaded by any other strategy (vertical line through this value entirely in the grey) and it can invade every other strategy (horizontal line through this value entirely in the white). Whatever strategy the population starts with, the population will evolve towards and end up at this value (note that the value is identical to the optimal obtained above, as predicted by the Mylius and Diekmann 1995 result). This can be demonstrated by selecting an arbitrary resident strategy, and then assuming a small mutation, and replacing the resident by the mutation if the mutation corresponds to an area in the white, and repeating.

Coexistence on ecological time-scales is defined by situations where two strategies can invade each other; corresponding to where both the matrix of λs plotted above and its mirror image are greater than 1. This is entirely uninteresting in this simple scenario (nothing can coexist except for exactly identical strategies) and is not therefore plotted.

While this provides the steps required, in the constant environment case where density dependence acts on seedling establishment, adaptive dynamics does not add much to a basic fitness analysis. More elaborate forms of density or frequency dependence and stochasticity can lead to more elaborate evolutionary trajectories reflected in more elaborate configurations of the plot (with strategies that might be locally but not globally convergent, etc).

7.4.8 Fitness in a stochastic environment

The evolutionary outcome of selection on timing of flowering can be rather different in stochastic environments. This section provides a quick introduction to estimating fitness in a stochastic environment, based on simulated data (since only one year of data are available in the Silwood data-set).

Briefly, stochastic analogues of the population rate of increase, stable population structure, etc. can be obtained by first, obtaining a sequence of environments by sampling a very large number of times from a list of IPMs representing the possible range of yearly environment types; and then

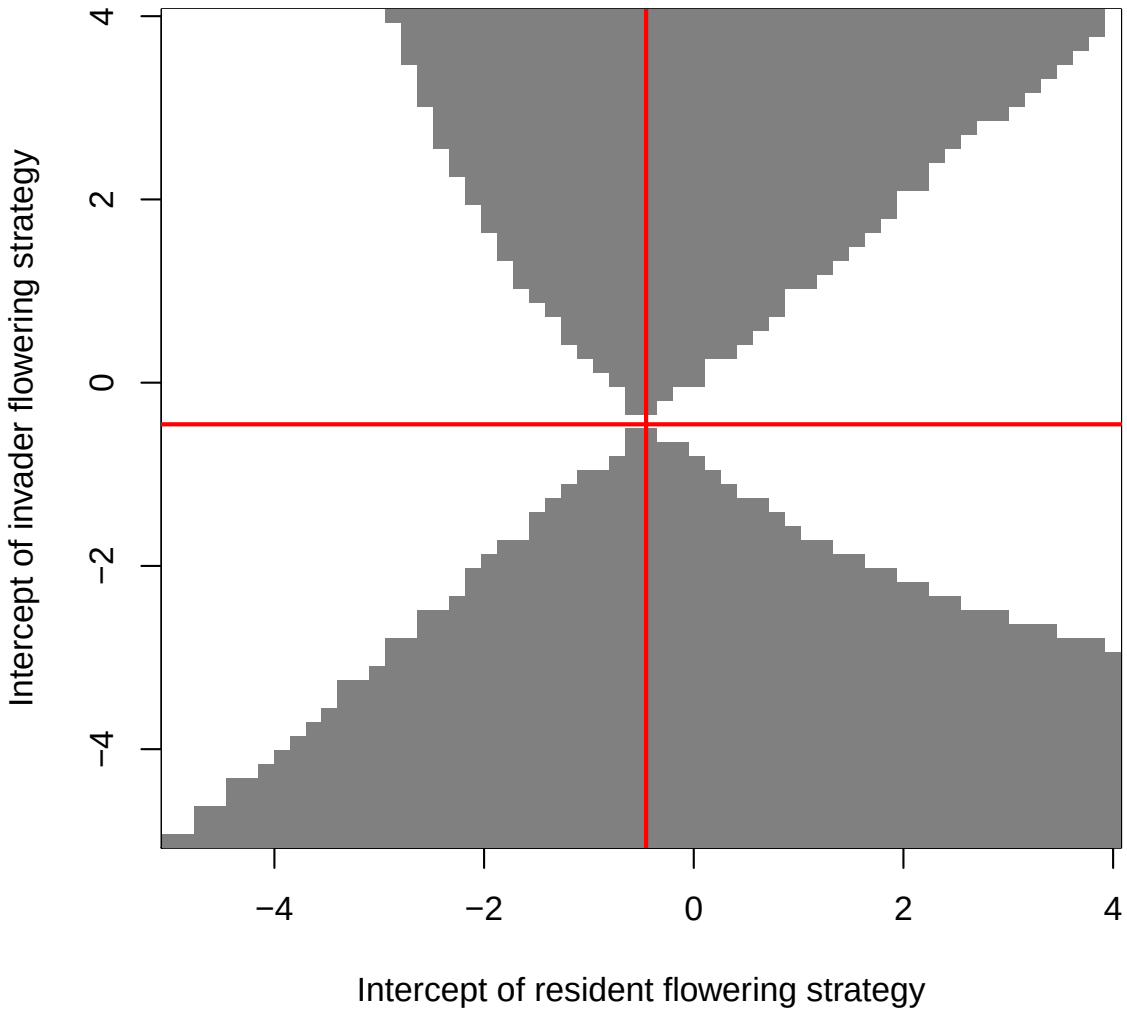

Figure 7.6: Adaptive dynamics surface: white areas indicate where the invader can successfully invade; grey where it will go extinct.

iterating a population through this sequence and storing population growth rate, stable population structure, etc; and relying on the weak ergodic theorem for convergence.

A function in IPMpack allows you to simulate a data-set based on parameters appropriate to *Carlina vulgaris*, used in many of the Rees et al. publications:

```
library(mvtnorm)
car<-simulateCarlina()
dff <- car$dataf
```

We can identify all the years for which we have data:

```
u.yr <- unique(dff$year)
```

and then fit growth, survival and fertility objects with different growth, survival and offspring size intercepts for every year:

```
dff$year <- as.factor(dff$year)
sv1<-makeSurvObj(dff,Formula=surv~size+year)
gr1<-makeGrowthObj(dff,Formula=sizeNext~size+year)
fv1 <- makeFecObj(dff, Formula = list(fec~size,flower~size),
                  Transform=c("none","none"),
                  Family=c("poisson","binomial"),
                  offspringSizeExplanatoryVariables="year")
```

(In reality, model selection should be used to identify the best models for each vital rate and define the appropriate formula). We then loop over the range of possible years to make list of `F` and `T` matrices for each year:

```
Pmat <- Fmat <- list()
for (j in 1:length(u.yr)) {

     #build current Pmat
     Pmat[[j]] <- makeIPMPmatrix(nBigMatrix = 50,
                                 minSize = 1, maxSize = 5,
                                 chosenCov = data.frame(year=as.factor(u.yr[j])),
                                 growObj=gr1, survObj=sv1,
                                 discreteTrans = 1,
                                 integrateType = "midpoint",
                                 correction = "constant")
     ##Take away the flowering guys!
     prob.not.flower <- 1-predict(fv1@fitFec[[2]],
                                 newdata=data.frame(size=Pmat[[j]]@meshpoints),
                                 type="response")
     Pmat[[j]] <- t(t(Pmat[[j]])*prob.not.flower)

     #build current Fmat
```

```
    Fmat[[j]] <- makeIPMFmatrix(nBigMatrix = 50,
                                minSize = 1, maxSize = 5,
                                chosenCov = data.frame(year=as.factor(u.yr[j])),
                                fecObj=fv1,
                                integrateType = "midpoint",
                                correction = "constant",
                                preCensus=FALSE,survObj=sv1,
                                growObj=gr1)
}
```

We can then estimate the stochastic growth rate, here assuming, as usual, density dependence in seedling establishment, so that the probability of establishment of seeds is defined by the ratio of the number of successful recruits in any year (recorded in the data-frame as `nSeedlings`) and the number of seeds produced by the total population in the previous year. There is a function to do this in IPMpack.

```
stochGrowthRateSampleList(listPmatrix=Pmat,
                          listFmatrix=Fmat,
                          nRunIn=1000, tMax=5000,
                          seedList=dff$nSeedlings[!duplicated(dff$year)],densDep=TRUE)
```

```
[1] -0.0001876126
```

The rate of invasion of a different strategy will be determined by the probability of establishment of the resident strategy as it moves through this sequence of environments (since we assume that the invader is too rare to initially affect establishment of itself or the resident). We therefore need to write function to store the list of environment types chosen, and the probability of establishment linked to a given resident strategy in this list of environments;

```
probEstSampleListDD <-  function (listPmatrix,
                                  listFmatrix,
                                  nRunIn, tMax,seedList) {
    require(MASS)
    #identify the number of possible environments
    nmatrices <- length(listPmatrix)
    #initiate the population
    nt <- rep(1, length(listPmatrix[[1]][, 1]))
    #set up storage
    Rt <- year.type <- pEst <- rep(NA, tMax)

    #loop over a suite of environments
    for (t in 1:tMax) {
        #pick out a year type
        year.type[t] <- sample(1:nmatrices, size = 1, replace = FALSE)
```

```
        #estimate the no of seeds that would have been produced in the previous year
        nseeds <- sum(listFmatrix[[year.type[t]]]%*%nt)
        #estimate the corresponding probability of establishment for this year (no of
        #microsites suiteable for seedling establishment available in the
        #the following year, divided by total number of seeds just produced)
        pEst[t] <- min(seedList[min(year.type[t]+1,nmatrices)]/nseeds,1)
        #project the current population one time-step forward
        nt1 <- (listPmatrix[[year.type[t]]]+
                pEst[t]*listFmatrix[[year.type[t]]])%*% nt
        #estimate the population growth rate
        sum.nt1 <- sum(nt1)
        Rt[t] <- log(sum.nt1)-log(sum(nt))
        #update the population size
        nt <- nt1
    }
    return(list(year.type=year.type,pEst=pEst, Rt=mean(Rt[nRunIn:tMax],na.rm=TRUE)))
}
```

and then we need a function to estimate the stochastic invasion rate in an identical suite of environments, where the probability of seed establishment is set by the resident strategy:

```
invasionRateSampleListDD <- function (listPmatrix, listFmatrix,
                                      nRunIn, tMax,year.type,pEst) {
    require(MASS)
    nmatrices <- length(listPmatrix)
    #initiate the population
    nt <- rep(1, length(listPmatrix[[1]][, 1]))
    #set up storage
    Rt <- rep(NA, tMax)
    for (t in 1:tMax) {
       #project the current population one time-step forward
       # imposing the residents probability of establishment
       nt1 <- (listPmatrix[[year.type[t]]]+
                pEst[t]*listFmatrix[[year.type[t]]])%*% nt
        #estimate the population growth rate
        sum.nt1 <- sum(nt1)
        Rt[t] <- log(sum.nt1)-log(sum(nt))
        #update the population size
        nt <- nt1
    }
    res <- mean(Rt[nRunIn:tMax], na.rm = TRUE)
```

```
    return(res)
}
```

Now we can estimate the invasion rate of a different flowering strategy into this population (say a smaller flowering strategy, so with a larger flowering intercept, say -11). First, get the vector of the suite of environments considered, and the appropriate probabilities of establishment for the chosen resident strategy:

```
res.cond <- probEstSampleListDD(listPmatrix=Pmat,
                                listFmatrix=Fmat,
                                nRunIn=1000, tMax=5000,
                                seedList=dff$nSeedlings[!duplicated(dff$year)])
```

Then alter the fecundity object, to reflect the invader's flowering function intercept:

```
fv1.i <- fv1
fv1.i@fitFec[[2]]$coefficients <- c(-11,fv1.i@fitFec[[2]]$coefficients[2])
```

and rebuild all the T and F matrices to reflect the invader's parameters:

```
Pmat.i <-  Fmat.i <- list()
for (j in 1:length(u.yr)) {
    #build current Pmat
    Pmat.i[[j]] <- makeIPMPmatrix(nBigMatrix = 50,
                                  minSize = 1, maxSize = 5,
                                  chosenCov = data.frame(year=as.factor(u.yr[j])),
                                  growObj=gr1, survObj=sv1,
                                    discreteTrans = 1,
                                  integrateType = "midpoint",
                                  correction = "constant")
    ##Take away the flowering individuals
    prob.not.flower <- 1-predict(fv1.i@fitFec[[2]],
                                 newdata=data.frame(size=Pmat.i[[j]]@meshpoints),
                                 type="response")
    Pmat.i[[j]] <- t(t(Pmat.i[[j]])*prob.not.flower)
    #build current Fmat
    Fmat.i[[j]] <- makeIPMFmatrix(nBigMatrix = 50,
                                    minSize = 1, maxSize = 5,
                                    chosenCov = data.frame(year=as.factor(u.yr[j])),
                                    fecObj=fv1.i,
                                    integrateType = "midpoint",
                                    correction = "constant",
                                    preCensus=FALSE,survObj=sv1,
                                    growObj=gr1)
```

```
}
```

Now estimate the stochastic invasion rate of this invader (with a flowering intercept of -11) into a population with the observed flowering intercept:

```
invasionRateSampleListDD(listPmatrix=Pmat.i,
                         listFmatrix=Fmat.i,
                         nRunIn=1000, tMax=5000,
                         year.type=res.cond$year.type,pEst=res.cond$pEst)
[1] -0.07197248
```

Note that you might not get this exact value - it will depend on the characteristics of the simulation; etc. You can use this approach to get the full invasion rate landscape for invading strategies as used in estimating the ESS above.

References

- Caswell. 2001. Matrix population models: analysis, construction and interpretation. 2nd ed. Sinauer, Sunderland, Massachusetts, USA.
- Childs, Rees, Rose, Grubb & Ellner. 2004. Evolution of size-dependent flowering in a variable environment: Construction and analysis of a stochastic integral projection model. Proc. Roy. Soc. Lond. Ser. B. 271, p471âĂŞ475.
- Childs, D.Z., Coulson, T.N., Pemberton, J.M., Clutton-Brock, T.H. & Rees, M. (2011). Predicting trait values and measuring selection in complex life histories: reproductive allocation decisions in Soay sheep. Ecology Letters, 14, 985-992.
- Klinkhamer, Meelis, de Jong & Weiner 1992. On the analysis of size-dependent reproductive output in plants. Functional Ecology. 6, p308-316.
- Metcalf, Rose & Rees. 2003. Evolutionary demography of monocarpic perennials. TREE 18, p471-479
- Metcalf, Rose & Rees. 2008. Seed predators and the evolutionarily stable flowering strategy in the invasive plant, *Carduus nutans* Evolutionary Ecology 23, p893-906
- Metcalf, Rees, Alexander & Rose. 2006. GrowthâĂŞsurvival trade-offs and allometries in rosette- forming perennials Functional Ecology 20, p217-225.
- Mylius & Diekmann 1995. On evolutionarily stable life-histories, optimization and the need to be specific about density-dependence. Oikos 74, p218âĂŞ224.
- Rees & Rose. 2002. Evolution of flowering strategies in *Oenothera glazioviana*: an integral projection model approach. Proc. Roy. Soc. Lond. Ser. B. 269, p1509-1515.

- Rees, M., Childs, D., Metcalf, J., Rose, K., Sheppard, A. & Grubb, P. (2006). Seed dormancy and delayed flowering in monocarpic plants: selective interactions in a stochastic environment. The American Naturalist, 168, E53-E71
- Rose, K.E., Louda, S.M. & Rees, M. 2005. Demographic and evolutionary impacts of native and invasive insect herbivores on *Cirsium canescens*. Ecology, 86, p453-465.
- Sletvold, N. 2005 Density-dependent growth and survival in a natural population of the facultative biennial *Digitalis purpurea* Journal of Ecology 93, p727-736.