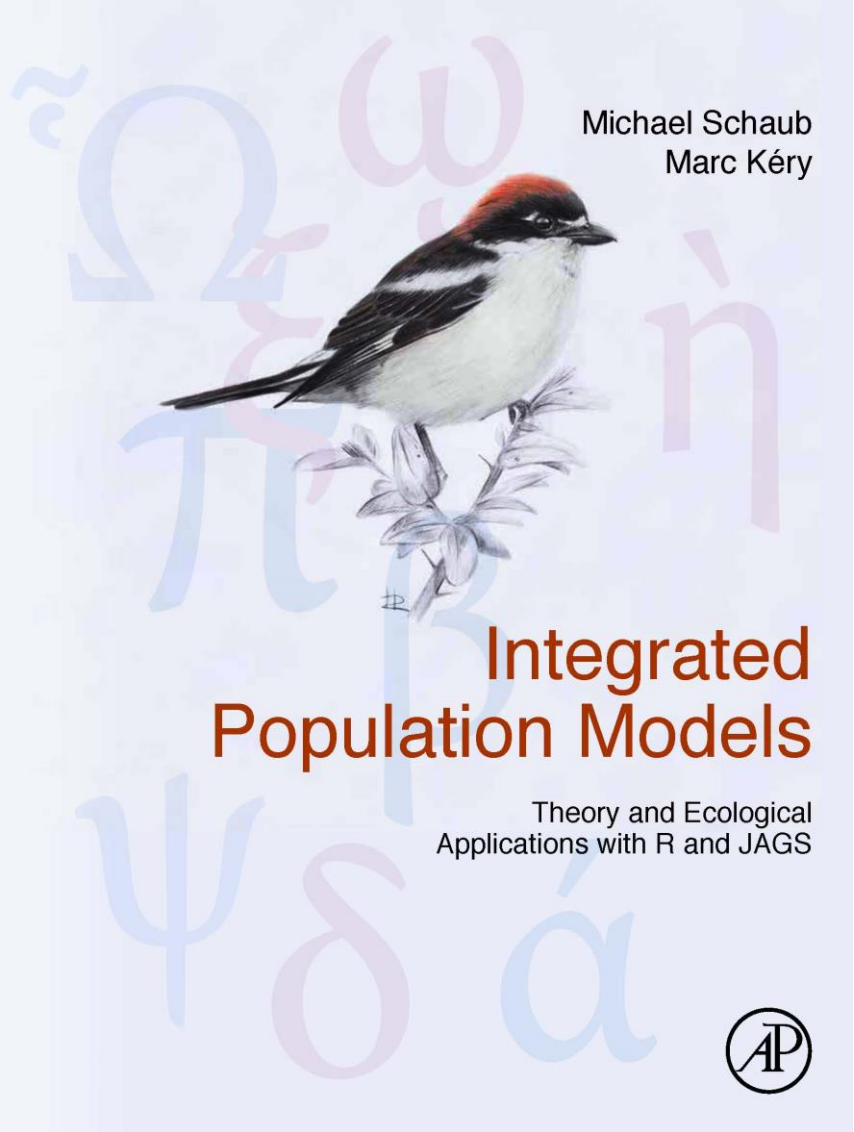
Solutions to the exercises in the book

Integrated Population Models.

Theory and ecological applications with R and JAGS



The solutions are organized by chapter. They contain code, results, figures, comments and sometimes some discussion as well. Note that you need to load the libraries `jagsUI` and `IPMbook` for most exercises, even though this is not always shown.

11 November 2021

Table of contents

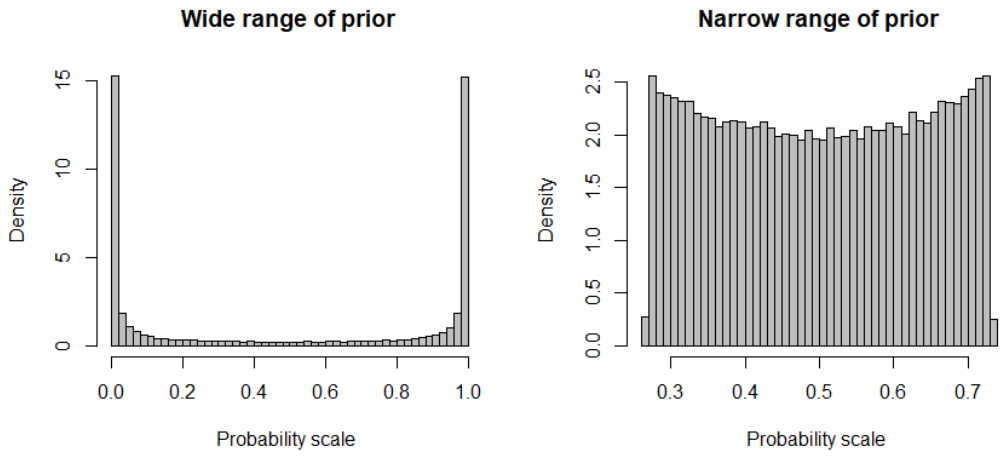
Chapter 2

Exercise 1

Convince yourself using simulation that a uniform prior for the intercept of a logistic regression is very informative about likely values of the intercept at the probability scale.

Solution. One possible solution is this: We start by generating random numbers from a uniform distribution on some range, e.g., from -10 to 10. Then we back-transform the values to the probability scale using the inverse logit function and produce a histogram of these values. We can vary the range of the uniform to see how this may affect the result.

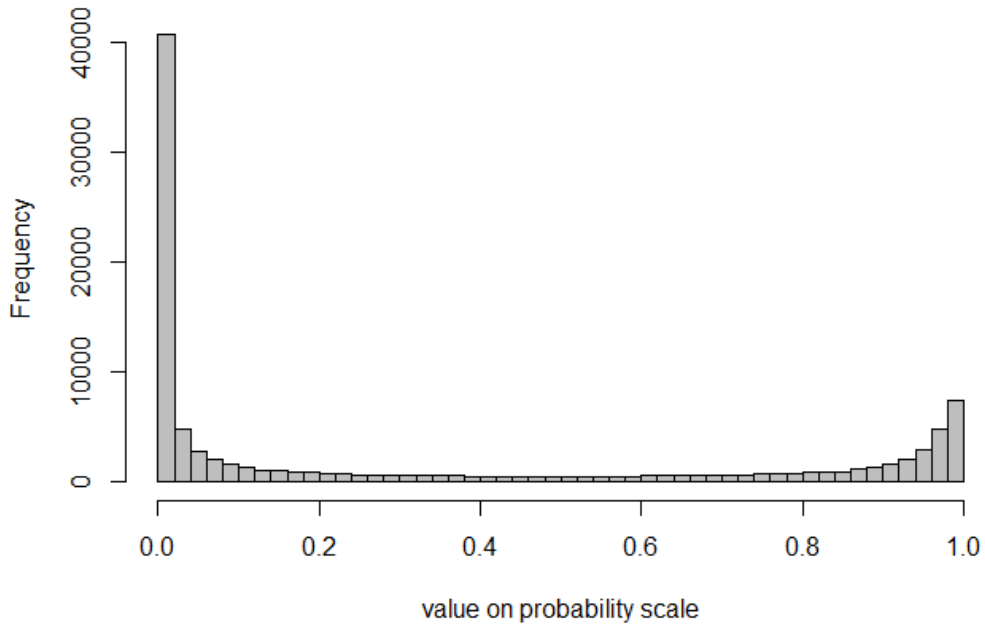
```
h1 <- runif(100000, -10, 10)  # This turns out to be very informative ...
h2 <- runif(100000, -1, 1)    # ... but this much less so.
j1 <- plogis(h1)
j2 <- plogis(h2)
par(mfrow=c(1,2))
hist(j1, nclass=50, col="grey", xlab="Probability scale", main="Wide range of
prior", freq=FALSE)
hist(j2, nclass=50, col="grey", xlab="Probability scale", main="Narrow range of
prior", freq=FALSE)
```



We clearly see that a uniform prior is not uninformative on the probability scale in the sense that any possible value is equally likely when we choose a usual very wide prior (that covers almost the entire 0-1 range of a probability). The shape of this prior becomes less extreme if the range of the uniform distribution is selected to be smaller, say from -1 to 1 as in our example. Yet, if the range gets too small, then not all possible values on the probability scale are included and this prior will then be informative, in that some possible values are excluded a priori. Furthermore, if the range of the uniform is not symmetric around 0, the mean of the values on the probability scale will deviate from 0.5, which might be undesired. This behavior is shown with the next graph.

```
h <- runif(100000, -10, 5)    # range not symmetric around 0
j <- plogis(h)
```

```
hist(j, nclass=50, col="grey", main=NA, xlab="Probability scale", freq=FALSE)
```



Now, in spite of the information carried by such priors, in practice they may be 'overwhelmed' by the data in moderate to large sample sizes. But nevertheless, we must know this potential problem and hence, care is necessary when defining priors for parameters that are expressed on different scales.

Exercise 2

In a tadpole experiment none of the 20 released individuals was seen. Analyze this data set both with Bayesian and maximum likelihood methods to estimate detection probability. Are the estimates and the SD or SE identical? If not, why are they different?

Solution. We start with a Bayesian analysis. We define the data and develop code for a logistic regression in JAGS and run the model.

```
# Data
N <- 20              # Number of tadpoles released initially
y <- 0               # Number of tadpoles detected later

jags.data <- list(N=N, y=y)

# Write JAGS model file
cat(file="model1.txt", "
model {
# Prior
p ~ dunif(0, 1)

# Likelihood
```

```
y ~ dbin(p, N)
}
")

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

# Parameters monitored
parameters <- c("p")

# MCMC settings
ni <- 50000; nb <- 10000; nc <- 3; nt <- 10; na <- 1000

# Call JAGS from R
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=FALSE)
print(out1, 3)

JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 50000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 10,
yielding 12000 total samples from the joint posterior.
MCMC ran for 0.01 minutes at time 2021-06-14 14:18:10.

          mean    sd  2.5%   50% 97.5% overlap0 f Rhat n.eff
p        0.045 0.043 0.001 0.032 0.161    FALSE 1    1 12000
deviance 1.904 1.876 0.050 1.319 7.012    FALSE 1    1 12000
```

The estimate of *p* and and its associated SD are obtained in no time.

Next, we analyze the same data set to produce the MLE in R using the function `glm`. We first load the data, which are now not aggregated, but consist of a data stream of 20 values of 0. Then we fit the intercept-only model.

```
# MLE solution
data <- rep(0, 20)
out2 <- glm(data ~ 1, family=binomial)
summary(out2)$coef

             Estimate Std. Error       z value  Pr(>|z|)
(Intercept) -25.56607    48299.1 -0.0005293281 0.9995777
```

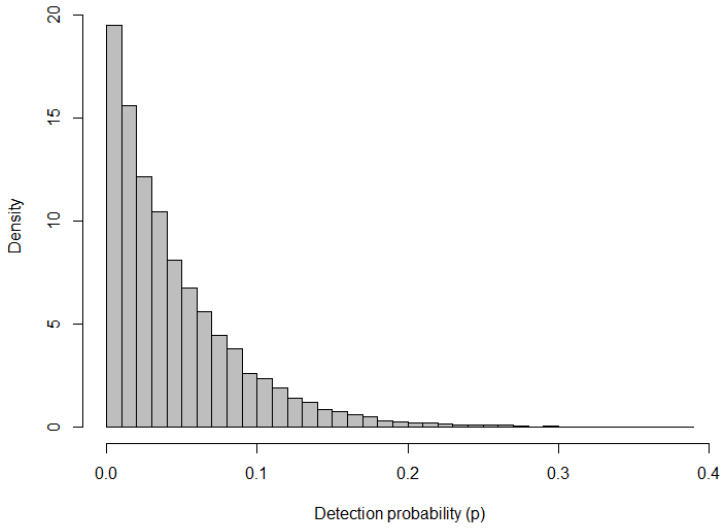
Naturally the estimate looks very different from that of the Bayesian analysis. However, recall that here the estimate is presented on the logit scale, and not on the probability scale as in our Bayesian analysis (this was our choice, and we could have parameterized the Bayesian model also on the logit-scale .... or simply computed the logit-scale value of detection probability as a derived quantity). To compare the estimates from the two analyses, we have to bring them to the same scale. One possibility is to calculate the back-transformed values from the MLE analysis using the function `predict`.

```
# To get values on prob scale
predict(out2, type="response", se.fit=TRUE)$fit[1]

           1
7.884924e-12

predict(out2, type="response", se.fit=TRUE)$se.fit[1]
```

```
          1
3.808347e-07
```

We see that the MLE is different from the Bayesian posterior mean; both the MLE and its SE look like being equal to zero. What we have here is a so-called boundary estimate in maximum likelihood, i.e., an estimate that falls at the boundary of the space where some parameter is defined (as 0 or 1 for a probability parameter). With no tadpole seen among 20 released, 0 is the valid MLE for detection probability *p*, but a SE of 0, meaning that there is no uncertainty, does not seem to be appropriate. In contrast, the Bayesian solution is based on the posterior distribution and that is the product of the likelihood AND the prior. And the prior says that all values between 0 and 1 are equally likely for *p*, and this, combined with the information in the data, then leads to a point estimate (in terms of the mean or the median) that is moved a little bit away from the boundary. In this way, boundary estimates do not appear in a Bayesian analysis when the posterior mean or median is used as a point estimate. Note, however, that also in the Bayesian analysis, the posterior mode (which is what corresponds to the MLE in an analysis with vague priors) lies at the boundary. We illustrate this here by producing a histogram of the posterior distribution.

```
hist(out1$sims.list$p, nclass=50, col="grey", main=NA, xlab="Detection
probability (p)", freq=FALSE)
```



Exercise 3

Repeat some of the described analyses with a variety of different prior distributions to examine the impacts of highly informative priors on posterior distributions.

Solution. We have solved this exercise for a single analysis – namely for the tadpole example in Exercise 2. In the solution above we used a flat, uniform prior (i.e., completely vague prior), and now we try two further priors. We imagine that both originate from previous studies where 4 and 40 tadpoles were released, of which 2 and 20, respectively, were then detected later, i.e., half of the released tadpoles in both cases. That is, in a sense these priors are both identical, in

that they say that our best guess at detection probability is that it is equal to 0.5. However, the prior that is based on more data (i.e., more tadpoles) is much stronger, as we will see below.

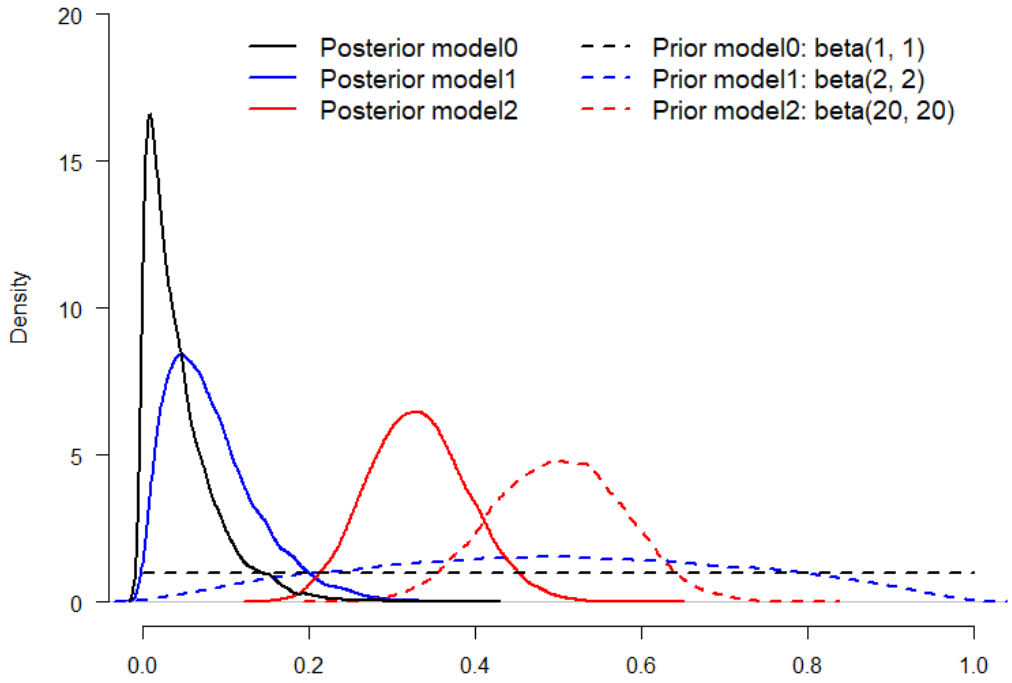
We use a beta distribution to incorporate that information from these previous hypothetical studies into our analysis as an informative prior. To run the model with the two priors (actually they are two separate analyses) in a single model fit in JAGS, we have to enter the data twice and name them y1 and y2. This is a "trick" with which we can in essence conduct multiple separate analyses within the same JAGS model statement.

```
# Data
N <- 20             # Number of tadpoles released initially
y <- 0              # Number of tadpoles detected later
jags.data <- list(N=N, y1=y, y2=y)

# Write JAGS model file
cat(file="model2.txt", "
model {
# Priors
# Prior 1
p1 ~ dbeta(2, 2)
# Prior 2
p2 ~ dbeta(20, 20)

# Likelihoods
y1 ~ dbin(p1, N)     # using informative prior 1
y2 ~ dbin(p2, N)     # using informative prior 2
}
")

# Initial values
inits <- function(){list(p1=runif(1))}

# Parameters monitored
parameters <- c("p1", "p2")

# MCMC settings
ni <- 50000; nb <- 10000; nc <- 3; nt <- 10; na <- 1000

# Call JAGS from R
out3 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=FALSE)
print(out3, 3)
```

```
JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 3 chains of 50000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 10,
yielding 12000 total samples from the joint posterior.
MCMC ran for 0.019 minutes at time 2021-06-14 15:11:37.

           mean    sd   2.5%    50%  97.5% overlap0 f Rhat n.eff
p1        0.084 0.055  0.011  0.072  0.221    FALSE 1    1 12000
p2        0.334 0.061  0.221  0.332  0.456    FALSE 1    1 12000
deviance 19.998 4.474 12.234 19.632 30.018    FALSE 1    1 12000
```

We see that the posterior means and SEs are very different from the earlier case where we had used the vague prior. We produce plots of the posteriors and the priors and compare them.

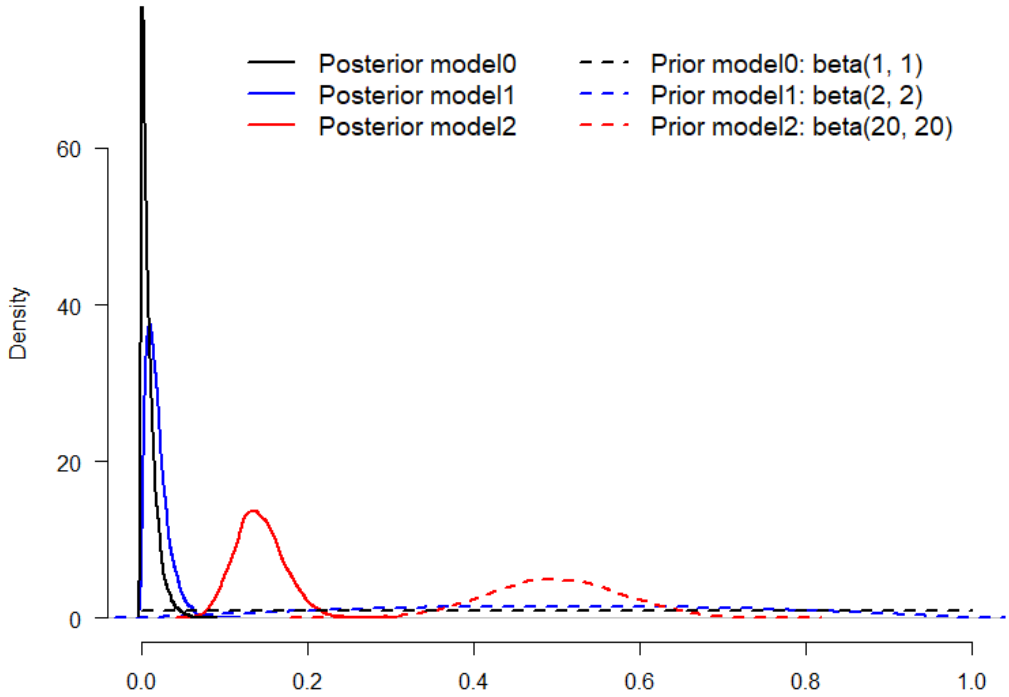
```
prior1 <- rbeta(12000, 2, 2)
prior2 <- rbeta(12000, 20, 20)

plot(density(out3$sims.list$p1), type="l", col="blue", lwd=2, main=NA,
xlim=c(0, 1), ylim=c(0, 20), axes=FALSE, xlab=NA)
axis(1)
axis(2, las=1)
lines(density(prior1), col="blue", lty=2, lwd=2)
lines(density(out3$sims.list$p2), col="red", lwd=2)
lines(density(prior2), col="red", lty=2, lwd=2)
lines(density(out1$sims.list$p), col="black", lwd=2)
segments(0, 1, 1, 1, col="black", lty=2, lwd=2)
legend(y=20, x=0.1, lty=c(1, 1, 1), lwd=c(2, 2, 2), col=c("black", "blue",
"red"), legend=c("Posterior model0", "Posterior model1", "Posterior model2"),
bty="n", cex=1.2)
legend(y=20, x=0.5, lty=c(2, 2, 2), lwd=c(2, 2, 2), col=c("black", "blue",
"red"), legend=c("Prior model0: beta(1, 1)", "Prior model1: beta(2, 2)", "Prior
model2: beta(20, 20)"), bty="n", cex=1.2)
```



The posteriors from the three analyses are different, and these differences illustrate the way in which the priors exert an influence in them (since this is the only difference among the three models). The more informative, i.e., the more peaked, the priors, the stronger the posterior will be affected.

If the data set is larger, then the impact of the same priors will decline proportionally. We illustrate this here by assuming that our data set was 5 times larger, hence that we had released 100 tadpoles and not a single one was detected later. The model to analyze this new data set and hence the three priors remains the same as before, so we can re-cycle it.

```
# New data for model 0
N <- 100           # Number of tadpoles released initially
y <- 0             # Number of tadpoles detected later
jags.data <- list(N=N, y=y)

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

# Parameters monitored
parameters <- c("p")

# MCMC settings
ni <- 50000; nb <- 10000; nc <- 3; nt <- 10; na <- 1000

# Call JAGS from R
out4 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=FALSE)

# New data for models 1 and 2
jags.data <- list(N=N, y1=y, y2=y)

# Initial values
inits <- function(){list(p1=runif(1))}

# Parameters monitored
parameters <- c("p1", "p2")

# MCMC settings
ni <- 50000; nb <- 10000; nc <- 3; nt <- 10; na <- 1000

# Call JAGS from R
out5 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=FALSE)


# Produce plots
prior1 <- rbeta(12000, 2, 2)
prior2 <- rbeta(12000, 20, 20)
plot(density(out5$sims.list$p1), type="l", col="blue", lwd=2, main=NA,
xlim=c(0, 1), ylim=c(0, 75), axes=FALSE, xlab=NA)
axis(1)
axis(2, las=1)
lines(density(prior1), col="blue", lty=2, lwd=2)
lines(density(out5$sims.list$p2), col="red", lwd=2)
lines(density(prior2), col="red", lty=2, lwd=2)
lines(density(out4$sims.list$p), col="black", lwd=2)
segments(0, 1, 1, 1, col="black", lty=2, lwd=2)
legend(y=75, x=0.1, lty=c(1, 1, 1), lwd=c(2, 2, 2), col=c("black", "blue",
"red"), legend=c("Posterior model0", "Posterior model1", "Posterior model2"),
bty="n", cex=1.2)
legend(y=75, x=0.5, lty=c(2, 2, 2), lwd=c(2, 2, 2), col=c("black", "blue",
"red"), legend=c("Prior model0: beta(1, 1)", "Prior model1: beta(2, 2)", "Prior
model2: beta(20, 20)"), bty="n", cex=1.2)
```



The posteriors under the three models become more similar than before, in particular the posteriors from model0 and model1 are now almost identical. Hence, we see how the impact of the priors on the posteriors, for a given data set, becomes smaller as the data set grows, and hence, the information in the data increases.

Chapter 3

Exercise 1

Compute the stable stage distribution of a woodchat shrike population when a post-breeding 'census' is applied. The demographic rates are the following: juvenile survival (0.30), adult survival (0.55), productivity of 1-year old females (1.3), productivity of adult females (1.8). Assume that the demographic rates are known without error and that they are constant over time. Ignore demographic stochasticity. Use JAGS for computation.

Solution: We start by drawing a life-cycle graph and by writing the population equations. As we write the matrix model for a post-breeding 'census', the youngest individuals in the population at the time of the 'census' are the fledglings. The other individuals are at least one year old and do not differ any more in terms of their demographic parameters. The simplified life-cycle graph is the same as shown in Fig. 3.7B. Based on that graph, we write the population equations:

$$N_{juv,t+1} = N_{juv,t} s_j f_1 + N_{ad,t} s_a f_a$$
$$N_{ad,t+1} = N_{juv,t} s_j + N_{ad,t} s_a$$

Next, we store the demographic values in a list, write the model in the BUGS language and run it. As there is no stochasticity in the model and this is a mere calculation exercise, the number of draws in the MCMC algorithm can be chosen to be minimal (we choose 5 here).

```
# Bundle data
jags.data <- list(sj=0.3, sa=0.55, f1=1.3, fa=1.8, T=5)

# Write JAGS model file
cat(file="model1.txt", "
model {
# Model for initial state
N[1,1] <- 1
N[2,1] <- 1

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

# Parameters monitored
parameters <- c("stable.stage", "N")

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

# Call JAGS (ART <1 min) and summarize results
out1 <- jags(jags.data, NULL, parameters, "model1.txt", n.adapt=na,
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, DIC=FALSE)
```

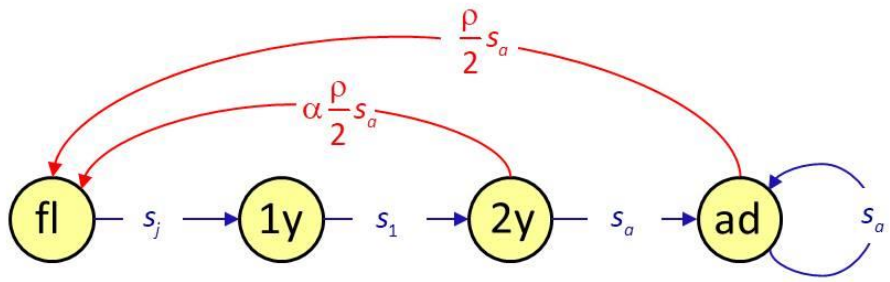
```
print(out1, 3)


JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 1 chains of 1 iterations,
adaptation = 0 iterations (sufficient),
burn-in = 0 iterations and thin rate = 1,
yielding 1 total samples from the joint posterior.
MCMC ran for 0 minutes at time 2021-10-11 13:02:32.

                 mean sd  2.5%   50% 97.5% overlap0 f
stable.stage[1] 0.611 NA 0.611 0.611 0.611    FALSE 1
stable.stage[2] 0.389 NA 0.389 0.389 0.389    FALSE 1
N[1,1]          1.000 NA 1.000 1.000 1.000    FALSE 1
N[2,1]          1.000 NA 1.000 1.000 1.000    FALSE 1
N[1,2]          1.380 NA 1.380 1.380 1.380    FALSE 1
N[2,2]          0.850 NA 0.850 0.850 0.850    FALSE 1
N[1,3]          1.380 NA 1.380 1.380 1.380    FALSE 1
N[2,3]          0.882 NA 0.882 0.882 0.882    FALSE 1
N[1,4]          1.411 NA 1.411 1.411 1.411    FALSE 1
N[2,4]          0.899 NA 0.899 0.899 0.899    FALSE 1
N[1,5]          1.440 NA 1.440 1.440 1.440    FALSE 1
N[2,5]          0.918 NA 0.918 0.918 0.918    FALSE 1
N[1,6]          1.470 NA 1.470 1.470 1.470    FALSE 1
N[2,6]          0.937 NA 0.937 0.937 0.937    FALSE 1
```

The stable stage distribution indicates that about 61% of the individuals are expected to be fledglings. The stable stage distribution is different compared to that obtained from formulating the population model for a pre-breeding 'census' (see Section 3.3.1), as expected.

Exercise 2

Compute the stochastic population growth rate of a common swift (*Apus apus*) population and its sensitivity to changes in all underlying demographic rates. We know the mean and the temporal variation (SD on natural scale) of the following demographic rates: juvenile survival: 0.4 (SD: 0.05); survival in the second year: 0.7 (SD: 0.02); adult (i.e., year 2+) survival 0.8 (SD: 0.01); probability to start reproduction at the age of 2 years: 0.5 (SD: 0.1); productivity per reproducing female: 1.7 (0.4). Use JAGS for the exercise.

Solution: We start by drawing a life-cycle graph and by writing down the population equations. We arbitrarily decide to use a post-breeding 'census', and therefore the youngest age class refers to the fledglings (aged 0 years).



The figure shows the postbreeding 'census' life-cycle graph of the common swift. Recruitment processes are depicted in red, pure survival processes in blue.

Population equations:

$$N_{fl,t+1} = N_{2y,t} s_{a,t} \alpha_t \frac{\rho_t}{2} + N_{ad,t} s_{a,t} \frac{\rho_t}{2}$$

$$N_{1y,t+1} = N_{fl,t} s_{j,t}$$

$$N_{2y,t+1} = N_{1y,t} s_{1,t}$$

$$N_{ad,t+1} = \left(N_{2y,t} + N_{ad,t}\right) s_{a,t}$$

The corresponding Leslie matrix is this:

$$\begin{bmatrix} 0 & 0 & s_a \alpha \frac{\rho}{2} & s_a \frac{\rho}{2} \\ s_j & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_a & s_a \end{bmatrix}_t$$

Next, we define the means and the temporal variances of the demographic rates, bundle them in a list and write the model in the BUGS language. The model is very similar to that in Section 3.4.3. The population model is of course different, since we model a different species, and here we directly use a beta distribution to generate annual probabilities of survival and of recruitment. In Section 3.4.3 we simulated these annual probabilities on the logit scale using a Normal distribution.

```
library(IPMbook)

# Define mean and temporal variability (SD) of the demographic parameters
mean.sj <- 0.4          # Mean juv. survival on probability scale
sd.sj.t <- 0.1          # Temporal variability of sj on probability scale
mean.s1 <- 0.7          # Mean 1y survival on probability scale
sd.s1.t <- 0.05         # Temporal variability of s1 on probability scale
mean.sa <- 0.8          # Mean ad survival on probability scale
sd.sa.t <- 0.02         # Temporal variability of sa on probability scale
mean.f <- 2             # Mean fecundity of 1y females on natural scale
sd.f.t <- 0.3           # Temporal variability of f1 on the natural scale
mean.alpha <- 0.5       # Mean recruitment on probability scale
sd.alpha.t <- 0.2       # Temp. variability of recruitment on the prob. scale

u <- 1000               # Define burn-in period for the stochastic simulation
T <- 10000              # Define the length of the stochastic simulation

# Bundle data
jags.data <- list(shape1.sj=getBeta2Par(mean.sj, sd.sj.t)[1],
shape2.sj=getBeta2Par(mean.sj, sd.sj.t)[2], shape1.s1=getBeta2Par(mean.s1,
sd.s1.t)[1], shape2.s1=getBeta2Par(mean.s1, sd.s1.t)[2],
shape1.sa=getBeta2Par(mean.sa, sd.sa.t)[1], shape2.sa=getBeta2Par(mean.sa,
sd.sa.t)[2], shape1.alpha=getBeta2Par(mean.alpha, sd.alpha.t)[1],
shape2.alpha=getBeta2Par(mean.alpha, sd.alpha.t)[2], mean.f=mean.f,
sd.f.t=sd.f.t, T=T+u, u=u)

# Write JAGS model file
cat(file="model2.txt", "
model {
# Use of RNG to accommodate temporal variability of demographic rates (process
variability)
for (t in 1:T){
  sj[t] ~ dbeta(shape1.sj, shape2.sj)
  s1[t] ~ dbeta(shape1.s1, shape2.s1)
  sa[t] ~ dbeta(shape1.sa, shape2.sa)
  alpha[t] ~ dbeta(shape1.alpha, shape2.alpha)
  f[t] ~ dnorm(mean.f, pow(sd.f.t, -2))T(0,)
}

# Model for initial state
for (s in 1:4){
  N[s,1] <- 1
}

# Loop over time
for (t in 1:T){
  # Population model
  N[1,t+1] <- sa[t] * f[t] / 2 * (N[3,t] * alpha[t] + N[4,t])
  N[2,t+1] <- sj[t] * N[1,t]
  N[3,t+1] <- s1[t] * N[2,t]
  N[4,t+1] <- sa[t] * (N[3,t] + N[4,t])

  # Annual growth rate on log scale
  r.annual[t] <- log(sum(N[,t+1])) - log(sum(N[,t]))
}

# Stochastic population growth rate
r <- mean(r.annual[u:T])
lambda <- exp(r)

# Calculate sensitivities of growth rate to changes in demographic rates
```

```
delta <- 0.001             # magnitude of perturbation
for (j in 1:5){
  N.star[1,1,j] <- 1
  N.star[2,1,j] <- 1
  N.star[3,1,j] <- 1
  N.star[4,1,j] <- 1
}

for (t in 1:T){
  # juvenile survival
  N.star[1,t+1,1] <- sa[t] * f[t] / 2 * (N.star[3,t,1] * alpha[t] +
N.star[4,t,1])
  N.star[2,t+1,1] <- (sj[t] + delta) * N.star[1,t,1]
  N.star[3,t+1,1] <- s1[t] * N.star[2,t,1]
  N.star[4,t+1,1] <- sa[t] * (N.star[3,t,1] + N.star[4,t,1])
  r.annual.star[t,1] <- log(sum(N.star[,t+1,1])) - log(sum(N.star[,t,1]))

  # 1y survival
  N.star[1,t+1,2] <- sa[t] * f[t] / 2 * (N.star[3,t,2] * alpha[t] +
N.star[4,t,2])
  N.star[2,t+1,2] <- sj[t] * N.star[1,t,2]
  N.star[3,t+1,2] <- (s1[t] + delta) * N.star[2,t,2]
  N.star[4,t+1,2] <- sa[t] * (N.star[3,t,2] + N.star[4,t,2])
  r.annual.star[t,2] <- log(sum(N.star[,t+1,2])) - log(sum(N.star[,t,2]))

  # adult survival
  N.star[1,t+1,3] <- (sa[t] + delta) * f[t] / 2 * (N.star[3,t,3] * alpha[t] +
N.star[4,t,3])
  N.star[2,t+1,3] <- sj[t] * N.star[1,t,3]
  N.star[3,t+1,3] <- s1[t] * N.star[2,t,3]
  N.star[4,t+1,3] <- (sa[t] + delta) * (N.star[3,t,3] + N.star[4,t,3])
  r.annual.star[t,3] <- log(sum(N.star[,t+1,3])) - log(sum(N.star[,t,3]))

  # alpha
  N.star[1,t+1,4] <- sa[t] * f[t] / 2 * (N.star[3,t,4] * (alpha[t] + delta) +
N.star[4,t,4])
  N.star[2,t+1,4] <- sj[t] * N.star[1,t,4]
  N.star[3,t+1,4] <- s1[t] * N.star[2,t,4]
  N.star[4,t+1,4] <- sa[t] * (N.star[3,t,4] + N.star[4,t,4])
  r.annual.star[t,4] <- log(sum(N.star[,t+1,4])) - log(sum(N.star[,t,4]))

  # productivity
  N.star[1,t+1,5] <- sa[t] * (f[t] + delta) / 2 * (N.star[3,t,5] * alpha[t] +
N.star[4,t,5])
  N.star[2,t+1,5] <- sj[t] * N.star[1,t,5]
  N.star[3,t+1,5] <- s1[t] * N.star[2,t,5]
  N.star[4,t+1,5] <- sa[t] * (N.star[3,t,5] + N.star[4,t,5])
  r.annual.star[t,5] <- log(sum(N.star[,t+1,5])) - log(sum(N.star[,t,5]))
}
for (j in 1:5){
  r.star[j] <- mean(r.annual.star[u:T,j])
  s[j] <- (exp(r.star[j])-lambda) / delta
}
}
")


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

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

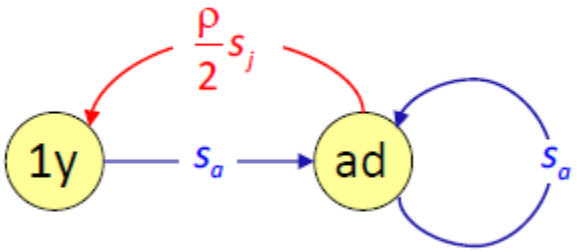
```
# Initial values
inits <- function(){list(sj=runif(T+u, 0, 1))}

# Call JAGS (ART <1 min) and summarize results
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.adapt=na,
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, DIC=FALSE)

print(out2, 4)

JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 1 chains of 1 iterations,
adaptation = 0 iterations (sufficient),
burn-in = 0 iterations and thin rate = 1,
yielding 1 total samples from the joint posterior.
MCMC ran for 0.549 minutes at time 2021-06-09 09:47:33.

        mean sd  2.5%   50% 97.5% overlap0 f
r      0.000 NA 0.000 0.000 0.000    FALSE 1
s[1]   0.338 NA 0.338 0.338 0.338    FALSE 1
s[2]   0.192 NA 0.192 0.192 0.192    FALSE 1
s[3]   0.915 NA 0.915 0.915 0.915    FALSE 1
s[4]   0.030 NA 0.030 0.030 0.030    FALSE 1
s[5]   0.067 NA 0.067 0.067 0.067    FALSE 1
lambda 1.000 NA 1.000 1.000 1.000    FALSE 1
```

The stochastic population growth rate is very close to 1 (remember that your results will slightly differ due to simulation variance), indicating that the population size remains stable. The stochastic sensitivities show that population growth is most sensitive to changes in adult survival and least to changes in recruitment (s[4]).

Exercise 3

Calculate the extinction probability of a lapwing (*Vanellus vanellus*) population and assess how much the uncertainty about the initial population size affects the outcome. We assume that lapwings start to reproduce when 2 years old, survival in the first year is 0.5, survival after their first birthday is 0.8 and productivity per female is 1.7. The size of the population is between 8 and 12 breeding pairs. Extinction probability shall be calculated with respect to 25 years and the demographic rates are assumed to be constant. Use JAGS for the calculations.

Solution: We start by drawing a life-cycle graph and by writing down the population equations. We decided to use a pre-breeding 'census' model, and thus the two stage classes are 1-year old individuals (1y) and individuals that are more than 1-year old (adults, ad). Below we show the life-cycle graph from which the population equations are derived. They include demographic stochasticity already, since this is a source of stochasticity that we need to include in our model for evaluating the extinction risk. By convention we use a Poisson distribution for the recruitment process and a Binomial distribution for the survival process.



The figure shows the prebreeding 'census' life-cycle graph of the lapwing. Recruitment processes are depicted in red, pure survival processes in blue.

Population equations:

$$N_{1y,t+1} \sim Poisson\left(N_{ad,t} s_j \frac{\rho}{2}\right)$$

$$N_{ad,t+1} \sim Binomial\left(N_{1y,t} + N_{ad,t}, s_a\right)$$

A challenge is how to include the uncertainty about the actual population size. We just know that the number of breeding pairs was between 8 and 12, but since lapwings only start to reproduce at the age of 2 years, there must be an unknown number of non-breeders in the population, which we must include in the population model as well. The best way to gauge the number of 1-year old individuals is to compute the stable stage distribution. Given the number of breeding pairs, this then allows us to calculate the number of 1-year old individuals.

If we know the number of individuals in each stage class, there are several possibilities to include the uncertainty about the number of breeding pairs (population size) in our model. A first approach is to use the maximum and the minimum numbers and calculate two extinction probabilities. These would define the ranges of the extinction probabilities, but each of them would assume a fixed value for the initial population size. A second possibility is to express the uncertainty in the population size using an appropriate distribution, from which we select values during the simulation. For example, we could use a Poisson distribution centered at the mean value. We will show both possibilities here.

First possibility: two scenarios with minimal and maximum values for the number of breeding pairs in the first year:

```
# Define mean of the demographic parameters
sj <- 0.4
sa <- 0.8
f <- 1.3

# Define the number of years with predictions
T <- 25

# Define initial stage-specific population sizes
N2 <- 8     # Number of breeding pairs (minimal value)

# Write Leslie matrix and calculate stable stage distribution
```

```
A <- matrix(c(0, sj*f/2, sa, sa), ncol=2, byrow=TRUE)
max(eigen(A)$values)
u <- which(eigen(A)$values == max(eigen(A)$values))
revec <- Re(eigen(A)$vectors[,u])
stable.stage <- revec/sum(revec)

# Calculate the number of 1y individuals based on stable stage distribution
N1 <- round(N2 * stable.stage[1] / stable.stage[2])

# Bundle data
jags.data <- list(sj=sj, sa=sa, f=f, T=T, N1=N1, N2=N2)

# Write JAGS model file
cat(file="model3.txt", "
model {
# Model for initial state
N[1,1] <- N1
N[2,1] <- N2

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

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

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

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

# Call JAGS (ART <1 min)
out3.min <- jags(jags.data, inits, parameters, "model3.txt", n.adapt=na,
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, DIC=FALSE)


# Define initial stage-specific population sizes
N2 <- 12    # Number of breeding pairs (maximal value)

# Calculate the number of 1y individuals based on stable stage distribution
N1 <- round(N2 * stable.stage[1] / stable.stage[2])

# Bundle data
jags.data <- list(sj=sj, sa=sa, f=f, T=T, N1=N1, N2=N2)

# Call JAGS (ART <1 min)
out3.max <- jags(jags.data, inits, parameters, "model3.txt", n.adapt=na,
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, DIC=FALSE)
```

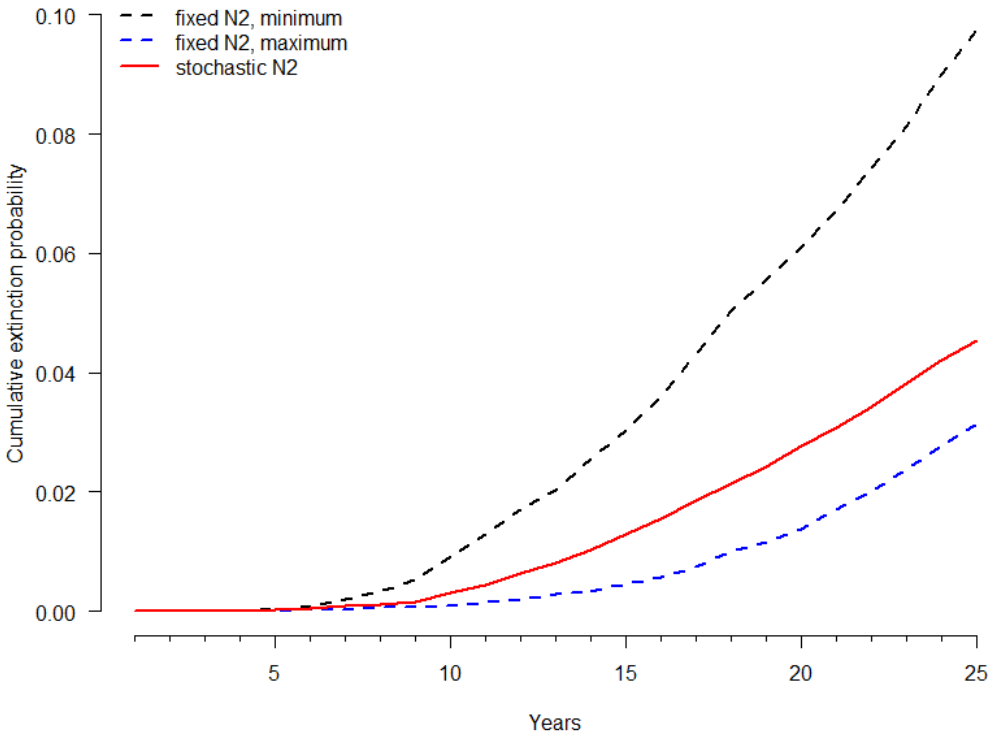
Second possibility: use the Poisson distribution centered on the mean value of the number of breeding pairs, i.e., (12+8)/2=10.

```
# Define initial stage-specific population sizes
N2 <- 10    # Number of breeding pairs (mean value)

# Write Leslie matrix and calculate stable stage distribution
A <- matrix(c(0, sj*f/2, sa, sa), ncol=2, byrow=TRUE)
max(eigen(A)$values)
u <- which(eigen(A)$values == max(eigen(A)$values))
revec <- Re(eigen(A)$vectors[,u])
stable.stage <- revec/sum(revec)

# Calculate the number of 1y individuals based on stable stage distribution
N1 <- round(N2 * stable.stage[1] / stable.stage[2])

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

# Write JAGS model file
cat(file="model4.txt", "
model {
# Model for initial state
N[1,1] ~ dpois(N1)
N[2,1] ~ dpois(N2)

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

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

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

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

# Call JAGS (ART <1 min)
out4 <- jags(jags.data, inits, parameters, "model4.txt", n.adapt=na,
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, DIC=FALSE)
```

To visualize the results, we produce a plot with the cumulative extinction probabilities over time under both scenarios.

```
par(mar=c(5, 5, 2, 1), las=1, mgp=c(3, 1, 0))
plot(out3.min$mean$extinct, type="l", lwd=2, lty=2, ylim=c(0, 0.1),
ylab="Cumulative extinction probability", xlab="Years", axes=FALSE)
axis(1)
axis(1, at=1:25, labels=NA, tcl=-0.25)
```

```
axis(2)
lines(out3.max$mean$extinct, lwd=2, lty=2, col="blue")
lines(out4$mean$extinct, lwd=2, col="red")
legend("topleft", legend=c("fixed N2, minimum", "fixed N2, maximum",
"stochastic N2"), lwd=rep(2, 3), lty=c(2, 2, 1), col=c("black","blue","red"),
bty="n")
```



Extinction probability is strongly affected by our assumptions about the number of breeding pairs in the first year. Obviously, extinction probability will be higher when the number of breeders in the first year is assumed to be smaller.

Chapter 4

Exercise 1

Given the mean (1.02) and temporal variability (log-scale variance = 0.02) of the population growth rate in an exponential model of growth, simulate one population trajectory of length 20 years starting from a population of size 50. Then, simulate a count data set from that population using a binomial distribution with a detection probability of 0.6, hence assuming that imperfect detection is the sole mechanism producing the observation error in the counts. Then, write and fit a state-space model that corresponds exactly to the data simulation and judge whether the parameters appear to be identifiable.

Solution. We simulate such a data set, analyze it, and inspect the estimated parameters. We stress that with a single realization from the simulated stochastic process we will not in general be able to assess whether a parameter is estimable or not from a given data set, but that in some cases this will nevertheless become clear. Hence, we first inspect results of a single data set, and then perform a more thorough simulation.

```
# Simulate population
nyears <- 20               # Number of years
N1 <- 50                   # Abundance at t = 1
mu.lam <- 1.02             # Mean of the distribution of lambda
sig2.lam <- 0.02           # Variance of the distribution of lambda
p <- 0.6                   # Detection probability

# Simulate true system state
N <- numeric(nyears)
N[1] <- N1                 # Set initial abundance
set.seed(1)                # Initialize the RNGs
lambda <- rnorm(nyears-1, mu.lam, sqrt(sig2.lam)) # Draw random lambdas
for (t in 1:(nyears-1)){
  N[t+1] <- lambda[t] * N[t] # Propagate population size forwards
}

# Simulate observations
y <- rbinom(nyears, round(N), p)
```

We then fit the data-generating model in JAGS.

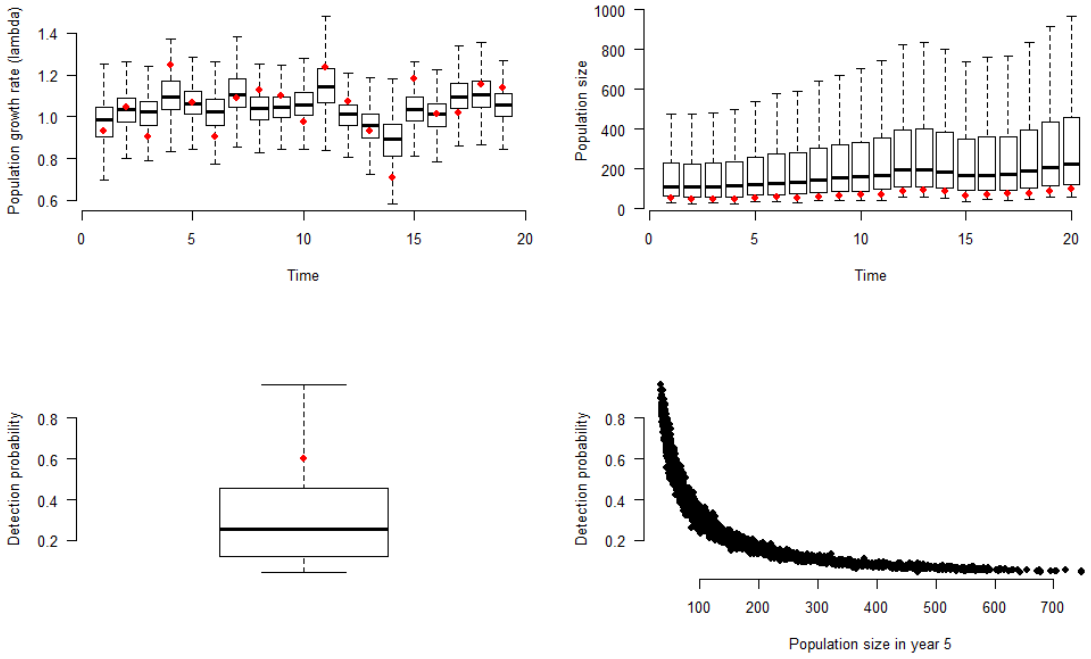
```
# Data bundle
jags.data <- list(y=y, T=length(y))

# Write JAGS model file
cat(file="model1.txt","
model {
# Priors and linear models
mu.lam ~ dunif(0, 10)            # Prior for mean growth rate
sig.lam ~ dunif(0, 1)            # Prior for sd of growth rate
sig2.lam <- pow(sig.lam, 2)
tau.lam <- pow(sig.lam, -2)
p ~ dunif(0, 1)

# Likelihood
# Model for the initial population size: uniform priors
```

```
N[1] ~ dunif(0, 500)

# Process model over time: our model of population dynamics
for (t in 1:(T-1)){
  lambda[t] ~ dnorm(mu.lam, tau.lam)
  N[t+1] <- N[t] * lambda[t]
}

# Observation process
for (t in 1:T) {
  y[t] ~ dbin(p, round(N[t]))
}
}
")

# Initial values
inits <- function(){list(sig.lam=runif(1, 0, 1))}

# Parameters monitored
parameters <- c("lambda", "mu.lam", "sig2.lam", "p", "N")

# MCMC settings
ni <- 100000; nb <- 10000; nc <- 3; nt <- 100; na <- 5000

# Call JAGS from R (ART <1 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)    # Not shown
print(out1, 3)


JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 1e+05 iterations,
adaptation = 5000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 100,
yielding 2700 total samples from the joint posterior.
MCMC ran for 4.326 minutes at time 2021-06-14 16:07:00.

              mean      sd   2.5%     50%   97.5% overlap0 f  Rhat n.eff
lambda[1]    0.973   0.112  0.737   0.984   1.177    FALSE 1 1.001  1575
lambda[2]    1.033   0.099  0.834   1.033   1.240    FALSE 1 1.003   698
[...output truncated...]
lambda[18]   1.114   0.095  0.946   1.104   1.329    FALSE 1 1.001  1888
lambda[19]   1.057   0.088  0.888   1.054   1.244    FALSE 1 1.002  2700
mu.lam       1.044   0.032  0.981   1.042   1.114    FALSE 1 1.001  2700
sig2.lam     0.017   0.015  0.000   0.014   0.056    FALSE 1 1.000  2700
p            0.309   0.214  0.058   0.255   0.795    FALSE 1 1.016   145
N[1]       159.541 124.860 37.006 110.139 467.094    FALSE 1 1.012   180
N[2]       157.450 126.844 32.166 108.392 470.500    FALSE 1 1.012   182
[...output truncated...]
N[19]      304.548 242.777 68.434 207.059 915.903    FALSE 1 1.009   221
N[20]      320.879 256.111 71.090 221.080 960.686    FALSE 1 1.010   214
deviance   120.114   8.745 97.368 121.391 133.640    FALSE 1 1.015   241
```

The model needs quite many iterations for the MCMC algorithm to converge. Next, we produce some plots to inspect the outcome. Basically, we compare the posteriors with the true values. If all works well, which in this case means a parameter can indeed be estimated from such a data set, then we expect that the true and the estimated values are similar.

```
# Produce plots: histograms of posteriors vs. true values
```

```
par(mfrow=c(2, 2))
boxplot(out1$sims.list$lambda, outline=FALSE, axes=FALSE, ylab="Population
growth rate (lambda)", xlab="Time")
points(x=1:19, y=lambda, col="red", pch=16)
axis(1); axis(2, las=1)

boxplot(out1$sims.list$N, outline=FALSE, axes=FALSE, ylab="Population size",
xlab="Time")
points(x=1:20, y=N, col="red", pch=16)
axis(1); axis(2, las=1)

boxplot(out1$sims.list$p, outline=FALSE, axes=FALSE, ylab="Detection
probability", xlab=NA)
points(x=1, y=p, col="red", pch=16)
axis(2, las=1)

plot(out1$sims.list$N[,5], out1$sims.list$p, pch=16, axes=FALSE,
ylab="Detection probability", xlab="Population size in year 5")
axis(1); axis(2, las=1)
```



The plots show histograms of the posteriors of the annual population growth rates, of the annual population sizes and of detection probability, along with the true values shown in red. We notice that population growth rates are quite well estimated – the true values are often close to their posterior means. Estimates of population sizes are less good, and clearly the posterior means are systematically larger than the true values. Moreover, detection probability is not well estimated, the posterior mean is lower than the true values. Finally, there is a strong negative sampling correlation between detection probability and population size (shown here for an example of population size in year 5). A strong negative sampling correlation between two parameters can be an indication that there are problems to separately estimate these parameters and that perhaps only the product of the two is estimable.

Hence, we tentatively conclude that abundance and detection are not separately estimable from such a data set when using this model. This is of course borne out in the literature (e.g.,

Knape and Koerner-Nievergelt 2015), where most authors agree that from such "single-visit data" abundance (or occupancy) and detection can only be separately estimated under strong parametric assumptions and that the models to do so may not be robust to assumption violations.

However, we emphasize that this "conclusion" is only a first stab at determining the estimability of the parameters in this model for the given data set. For a more complete assessment of the identifiability (which is the opposite to intrinsic non-identifiability) or the estimability (the opposite of what is also called extrinsic non-identifiability) of a model for given data set, we must either use formal methods (see the recent book by Cole 2020) or else use simulations, where we repeat the data generation/data analysis cycle many times and then look at the average behavior of the estimators. In practice, simulation is the most flexible and powerful method for a non-statistician to gain insight into identifiability questions (see Chapter 3 in Kéry and Royle 2016). Therefore, we next illustrate this with the given example.

To run a proper simulation, we will repeatedly simulate a data set, analyze it, and save the estimates. We can then compare the posterior means with the true values and assess estimator bias. If you do that for this example, you will see that the parameters (population sizes and detection probability, but not population growth rates) are strongly biased, suggesting that the model does not "work" properly for the estimation of population size and detection probability from this type of data. You might also change the values used for the data simulation to understand whether the problem is restricted to some specific parameter values (or combination thereof) or whether these problems are endemic. We don't show this latter here, i.e., we only conduct the simulation with the same data-generating values specified above.

For simulations it is often useful to package some code in functions. This then makes it easier to execute several R commands all at once, and, if some constants used in the data simulation are made arguments of the function, to change the settings of a simulation more readily. Normally we would document the function arguments in more detail, but here we won't do this because we're lazy. In the function definition that follows here we give all function arguments default values, which will be used when the function is called, unless they are set at some other value. For fun we also add graphical function output. Note that we had to drop the call to `set.seed()`, since otherwise the simulation function would have been pointless.

```
# Package the data simulation code into a function
# ----------- Start function definition in R --------------
popsim.fn <- function(nyears=20, N1=50, mu.lam=1.02, sig2.lam=0.02, p=0.6){
  # Simulate true system state
  N <- numeric(nyears)
  N[1] <- N1                # Set initial abundance
  lambda <- rnorm(nyears-1, mu.lam, sqrt(sig2.lam)) # Draw random lambdas
  for (t in 1:(nyears-1)){
    N[t+1] <- lambda[t] * N[t] # Propagate population size forwards
  }

  # Simulate observations
  y <- rbinom(nyears, round(N), p)

  # Graphical output
  par(cex.axis=1.5, cex.lab=1.5)
  ylim <- c(0.8, 1.3) * range(c(N, y))
  plot(1:nyears, N, xlab='Years', ylab='Number', pch=16, cex=1.5, col='red',
frame=FALSE, ylim=ylim, type='b')
  points(1:nyears, y, pch=16, cex=1.5, col='blue', type='b')
  legend('topleft', legend=c('True N', 'Observed y'), pch=16, col=c('red',
'blue'), bty='n', cex=1.5)
```

```
  # Numerical output
  return(list(nyears=nyears, N1=N1, mu.lam=mu.lam, sig2.lam=sig2.lam, p=p, N=N,
lambda=lambda, y=y))
}# ----------- End of function definition in R --------------
```

Once we execute this code in R we have defined the function and then can use it.

```
str(data <- popsim.fn())   # call with default function arguments
```

PLEASE execute this function a couple of times now .... it is just breath-taking to see how radically different multiple realizations from one and the same stochastic process can be!

Next are some examples of the use of the function when default function arguments are changed.

```
str(data <- popsim.fn(nyears=100))    # More years
str(data <- popsim.fn(N1=10))         # Smaller initial pop. size
str(data <- popsim.fn(mu.lam=0.96))   # Decline (on average)
str(data <- popsim.fn(sig2.lam=0.1)) # More variability in lambda
str(data <- popsim.fn(p=0.9))         # Higher detection prob.
```

Once we have marveled at the power of our new function, we are ready to use it in a full-blown simulation assessment of the estimability of the parameters in our model. For this, we can re-cycle several parts of the analysis from above; for instance, there is of course no need to redefine the model in the BUGS language at every iteration of the simulation.

But first, we need to determine the number of simulation replicates we desire and also create a data structure to save the results (i.e., both the true values and the estimates) at every iteration of the simulation. This then enables us to compare the truth (which, due to simulation variability, will be different at each replicate, at least for some quantities such as *N*). To save the simulation data we use a list, while for the results, we simply save the marginal posterior summary table produced by `jags`. The code below requires you to have fit the model once and produced such a summary table beforehand.

```
# Determine sample size and create R structures for saving
simrep <- 50   # For a 'real' simulation should perhaps use 100 or 1000
truth <- list()
estimates <- array(NA, dim=c(dim(out1$summary), simrep))
dimnames(estimates)[1:2] <- dimnames(out1$summary)
```

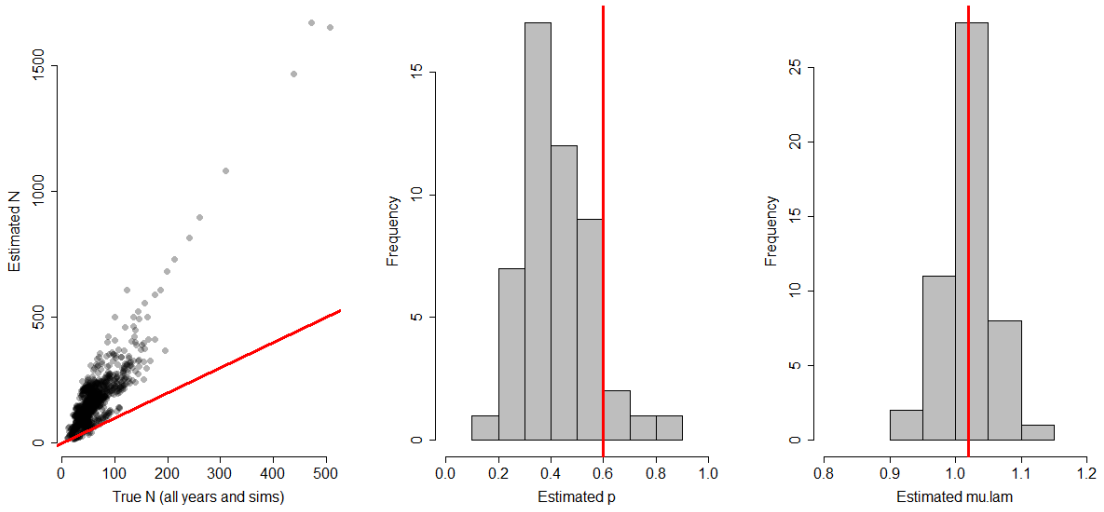
Now we execute the data simulation/data analysis code 'simrep' times by embedding it in a loop.

```
# Launch simulation
for(s in 1:simrep){
  cat(paste('\n**** Sim', s, '****\n'))
  # (1) Simulate a data set
  data <- popsim.fn()
  jags.data <- list(y=data$y, T=data$nyears)

  # (2) Fit the model
  # Initial values .... same as before
  # Parameters monitored .... also same as before
  # MCMC settings ... ditto
  # Call JAGS from R (ART 1 min)
```

```
  out1x <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
    n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)

  # (3) Save both the truth and the estimates
  truth[[s]] <- data
  estimates[,,s] <- out1x$summary
}
```

We do a simple check for greater generality (i.e., for multiple replications of the simulated stochastic process) of our conclusions based on the analysis of just one single data set. For this, we just plot the estimated against the true values of the latent abundance states *N* and produce a histogram of the point estimates of detection probability.

```
# Extract values of N and their estimates (posterior means)
Ntrue <- NULL
for(s in 1:simrep){
  Ntrue <- c(Ntrue, truth[[s]]$N)
}
Nest <- c(estimates[23:42,1,])

# Compare estimates with known truth
par(mfrow=c(1, 3), mar=c(5, 5, 4, 3), cex.axis=1.5, cex.lab=1.5, las=0)
ylim <- range(c(Nest, Ntrue))
plot(Ntrue, Nest, xlab='True N (all years and sims)', ylab='Estimated N',
pch=16, cex=1.5, col=rgb(0, 0, 0, 0.3), frame=FALSE, ylim=ylim)
abline(0, 1, lwd=3, col='red')
hist(estimates[22,1,], mai='', col='grey', xlab='Estimated p', xlim=c(0, 1))
abline(v=data$p, lwd=3, col='red')
hist(estimates[20,1,], mai='', col='grey', xlab='Estimated mu.lam', xlim=c(0.8,
1.2))
abline(v=data$mu.lam, lwd=3, col='red')
```



Thus, we confirm our first impression from the analysis of a single data set: there is something fundamentally wrong with the estimates of *N* and *p* in this model and we here assume that this is due to lack of identifiability of these parameters under this model and for this type of data. Yet, the mean population growth rate turns out to be estimated without bias, and hence the model can be used for making inference about population trends.

Exercise 2

Using the wryneck nest survival data (`data(wryneck)` in the `IPMbook` package) fit a model that includes an additive effect of the hatching date on daily nest survival in addition to the age effects (days since hatching). Do early nests have higher survival than late nests?

Solution. For a start we take the code from Section 4.4.6 and adapt it to our needs. We must add a linear term in the model for the daily nest survival probability, such that the latter changes as a function of the hatching date. Hatching date must be added to the data bundle. We then run the model and produce a graph of nest survival (i.e., of the product of daily nest survivals over the entire nestling period) and do this for different hatching dates.

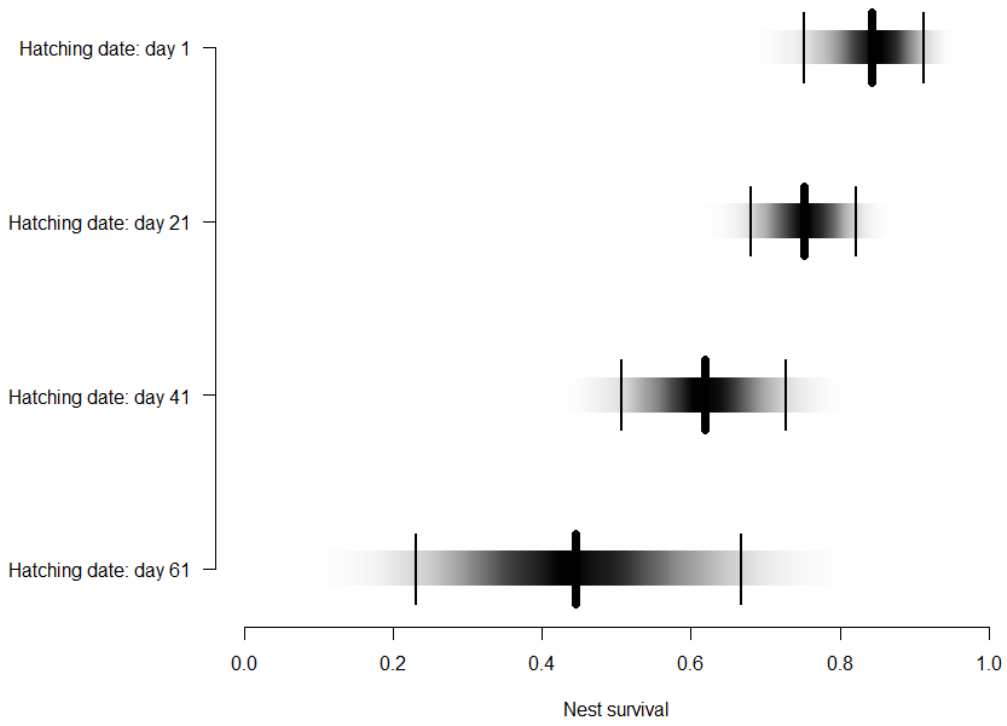
```
# Load the wryneck data
library(IPMbook)
data(wryneck)

# Identify failed broods
fail <- which(wryneck$x==0)

# Create encounter histories
y <- matrix(NA, nrow=length(wryneck$f), ncol=max(wryneck$k))
for (i in 1:length(wryneck$f)){
  y[i,wryneck$f[i]] <- 1
  y[i,wryneck$j[i]] <- 1
}
for (i in 1:length(fail)){
  y[fail[i],wryneck$k[fail[i]]] <- 0
}

# Compute hatching date (computed from the detection date and the age of the
nestling)
hatch.date <- wryneck$f - wryneck$age + 1

# Bundle data
jags.data <- with(wryneck, list(y=y, f=f, k=k, n.nest=nrow(y), T=21, age=age,
date=hatch.date))

# Write JAGS model file
cat(file="model2.txt", "
model {
# Priors and linear models
for (i in 1:n.nest){
  for (t in f[i]:(k[i]-1)){
    phi[i,t] <- phia[i,age[i] + t - f[i]]
  } #t
} #i

for (i in 1:n.nest){
  for (a in 1:T){
    phia[i,a] <- ilogit(alpha + beta[1] * a + beta[2] * date[i])
  } #a
} #i
alpha ~ dnorm(0, 0.001)
beta[1] ~ dnorm(0, 0.001)
beta[2] ~ dnorm(0, 0.001)
```

```
# Likelihood
for (i in 1:n.nest){
  for (t in (f[i]+1):k[i]){
    y[i,t] ~ dbern(phi[i,t-1] * y[i,t-1])
  } #t
} #i

# Derived parameters
# Daily nest survival for different hatching dates
for (a in 1:T){
  sur[1,a] <- ilogit(alpha + beta[1] * a + beta[2] * 1)  # Hatch day 1
  sur[2,a] <- ilogit(alpha + beta[1] * a + beta[2] * 21) # Hatch day 21
  sur[3,a] <- ilogit(alpha + beta[1] * a + beta[2] * 41) # Hatch day 41
  sur[4,a] <- ilogit(alpha + beta[1] * a + beta[2] * 61) # Hatch day 61
}

# Compute nest success (product of daily nest survivals)
# For hatching day 1
nu[1] <- prod(sur[1,1:T])
# For hatching day 21
nu[2] <- prod(sur[2,1:T])
# For hatching day 41
nu[3] <- prod(sur[3,1:T])
# For hatching day 61
nu[4] <- prod(sur[4,1:T])
}
")

# Initial values
inits <- function(){list(alpha=runif(1, 4, 5), beta=runif(2, 0, 0.1))}

# Parameters monitored
parameters <- c("nu", "alpha", "beta", "sur")

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

# Call JAGS from R (ART 2 min) and check convergence
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out2)       # Not shown
print(out2, 3)

JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran in parallel for 1.943 minutes at time 2021-10-13 14:34:12.

             mean     sd   2.5%     50%   97.5% overlap0     f  Rhat n.eff
nu[1]       0.842  0.041  0.752   0.846   0.912    FALSE 1.000 1.009   236
nu[2]       0.753  0.036  0.680   0.754   0.821    FALSE 1.000 1.004   553
nu[3]       0.619  0.056  0.506   0.619   0.728    FALSE 1.000 1.001  3084
nu[4]       0.446  0.110  0.230   0.443   0.667    FALSE 1.000 1.004   513
alpha       4.145  0.406  3.363   4.141   4.932    FALSE 1.000 1.011   193
beta[1]     0.076  0.032  0.016   0.076   0.140    FALSE 0.993 1.004   511
beta[2]    -0.027  0.009 -0.044  -0.027  -0.009    FALSE 0.998 1.009   246
sur[1,1]    0.984  0.006  0.970   0.985   0.993    FALSE 1.000 1.009   224
sur[2,1]    0.974  0.007  0.957   0.975   0.986    FALSE 1.000 1.005   369
sur[3,1]    0.957  0.012  0.928   0.958   0.977    FALSE 1.000 1.002  1448
sur[4,1]    0.927  0.027  0.865   0.930   0.968    FALSE 1.000 1.001  2095
```

```
sur[1,2]     0.985  0.005  0.973   0.986   0.993    FALSE 1.000 1.010   214
sur[2,2]     0.976  0.006  0.962   0.977   0.986    FALSE 1.000 1.006   351
sur[3,2]     0.960  0.011  0.936   0.961   0.977    FALSE 1.000 1.001  1640
sur[4,2]     0.932  0.024  0.878   0.935   0.969    FALSE 1.000 1.001  1947
[...output truncated...]
sur[1,21]    0.996  0.002  0.992   0.997   0.999    FALSE 1.000 1.002  1255
sur[2,21]    0.994  0.003  0.988   0.994   0.998    FALSE 1.000 1.002   993
sur[3,21]    0.990  0.004  0.979   0.990   0.996    FALSE 1.000 1.007   408
sur[4,21]    0.982  0.009  0.959   0.984   0.994    FALSE 1.000 1.012   270
deviance   107.004 21.082 68.532 106.342 150.656    FALSE 1.000 1.000  4339
```

The model converges swiftly. Our focal parameter `beta[2]` is clearly negative, indicating that daily nest survival declines as a function of hatch date over the season. We next produce plots of the marginal posteriors for total nest success in relation to the hatching date (for 4 different levels).

```
# Produce plot of nest survival
library(denstrip)
par(mar=c(4.5, 10, 1, 1))
plot(0, ylim=c(0.8, 4.2), xlim=c(0, 1), axes=F, pch=NA, xlab="Nest survival",
ylab=NA)
denstrip(out2$sims.list$nu[,1], at=4, ticks=c(out2$mean$nu[1], out2$q2.5$nu[1],
out2$q97.5$nu[1]), twd=c(7,2.5,2.5), tlen=c(2,2,2), width=1/5)
denstrip(out2$sims.list$nu[,2], at=3, ticks=c(out2$mean$nu[2], out2$q2.5$nu[2],
out2$q97.5$nu[2]), twd=c(7,2.5,2.5), tlen=c(2,2,2), width=1/5)
denstrip(out2$sims.list$nu[,3], at=2, ticks=c(out2$mean$nu[3], out2$q2.5$nu[3],
out2$q97.5$nu[3]), twd=c(7,2.5,2.5), tlen=c(2,2,2), width=1/5)
denstrip(out2$sims.list$nu[,4], at=1, ticks=c(out2$mean$nu[4], out2$q2.5$nu[4],
out2$q97.5$nu[4]), twd=c(7,2.5,2.5), tlen=c(2,2,2), width=1/5)
axis(1)
axis(2, las=1, at=c(4,3,2,1), labels=c("Hatching date: day 1", " Hatching date:
day 21", " Hatching date: day 41", " Hatching date: day 61"))
```



Thus, nest survival declines over the season as a function of the hatch date of a brood. A brood that hatches two months later than the earliest brood of a year (i.e., on day 61) is estimated to have a nest survival that is only about half of that of the earliest broods.

Exercise 3

Use the woodchat shrike capture-recapture data (`data(woodchat5)` in the `IPMbook` package) and first fit a model with age-dependent survival and age-independent recapture probability. Then, fit a model in which survival is also a function of the population counts, i.e., in which you add a form of density dependence. Is there any evidence that survival changes as a function of this index of population size?

Solution. We start by fitting a model without density dependence to have a reference model. We load the data and run the model.

```
# Load data
library(IPMbook)
data(woodchat5)

# Compute m-arrays
marr <- marrayAge(woodchat5$ch, woodchat5$age)

# Bundle data and produce data overview
jags.data <- list(marr.j=marr[,,1], marr.a=marr[,,2], n.occasions=dim(marr)[2],
rel.j=rowSums(marr[,,1]), rel.a=rowSums(marr[,,2]))

# Write JAGS model file
cat(file="model3.txt", "
model {
# Priors and linear models
mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

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

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

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

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

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

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

# Call JAGS (ART <1 min), check convergence and summarize posteriors
out3 <- jags(jags.data, inits, parameters, "model3.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na)

JAGS output for model 'model3.txt', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran for 0.152 minutes at time 2021-10-13 14:42:12.

             mean    sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj     0.305 0.016   0.274   0.305   0.336    FALSE 1 1.003   672
mean.sa     0.542 0.014   0.515   0.542   0.570    FALSE 1 1.000  6000
mean.p      0.603 0.021   0.561   0.603   0.643    FALSE 1 1.001  1816
deviance 460.230 2.386 457.502 459.629 466.480    FALSE 1 1.001  6000
```

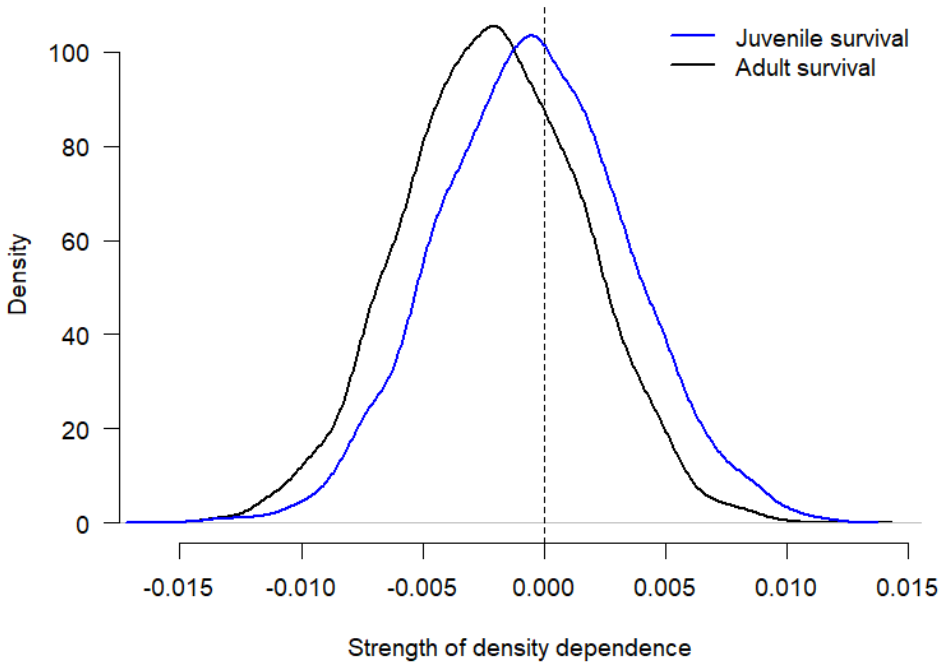
Next, we fit the model with our formulation of density dependence in survival. We define a linear model in which annual survival probability is a logit-linear function of our population size index (i.e., the counts). Under the hypothesis of density dependence, we would expect a negative relationship in this regression. We need to add the index for population size into our data bundle. Actually, we will work with the mean-centered population counts since this enhances rates of convergence of the MCMC algorithm.

```
# Bundle data and produce data overview
jags.data <- list(marr.j=marr[,,1], marr.a=marr[,,2], n.occasions=dim(marr)[2],
rel.j=rowSums(marr[,,1]), rel.a=rowSums(marr[,,2]), N=woodchat5$count -
mean(woodchat5$count))

# Write JAGS model file
cat(file="model4.txt", "
model {
# Priors and linear models
for (i in 1:2){
  alpha[i] ~ dnorm(0, 0.01)
```

```
  beta[i] ~ dnorm(0, 0.01)
}
mean.p ~ dunif(0, 1)

# Linear model for survival on population size (linear on logit scale)
for (t in 1:(n.occasions-1)){
  sj[t] <- ilogit(alpha[1] + beta[1] * N[t])
  sa[t] <- ilogit(alpha[2] + beta[2] * N[t])
  p[t] <- mean.p
}

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

# Initial values
inits <- function(){list(alpha=runif(2, -1, 1))}

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

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

# Call JAGS, check convergence and summarize posteriors
out4 <- jags(jags.data, inits, parameters, "model4.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na)
print(out4)

JAGS output for model 'model4.txt', generated by jagsUI.
Estimates based on 3 chains of 3000 iterations,
adaptation = 1000 iterations (sufficient),
burn-in = 1000 iterations and thin rate = 1,
yielding 6000 total samples from the joint posterior.
MCMC ran for 0.259 minutes at time 2021-10-13 14:43:44.
```

```
            mean    sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
alpha[1]  -0.828 0.076  -0.974  -0.828  -0.674    FALSE 1.000 1.001  5530
alpha[2]   0.171 0.059   0.053   0.172   0.288    FALSE 0.998 1.001  1320
beta[1]    0.000 0.004  -0.008   0.000   0.007     TRUE 0.552 1.001  2257
beta[2]   -0.002 0.004  -0.009  -0.002   0.005     TRUE 0.710 1.000  3091
sj[1]      0.309 0.036   0.242   0.308   0.381    FALSE 1.000 1.002  1518
sj[2]      0.305 0.018   0.271   0.305   0.343    FALSE 1.000 1.002  2060
sj[3]      0.304 0.016   0.274   0.304   0.337    FALSE 1.000 1.000  6000
[...output truncated...]
sj[17]     0.303 0.020   0.266   0.303   0.343    FALSE 1.000 1.000  6000
sj[18]     0.302 0.025   0.256   0.302   0.352    FALSE 1.000 1.000  6000
sj[19]     0.303 0.022   0.260   0.302   0.348    FALSE 1.000 1.000  6000
sa[1]      0.562 0.038   0.487   0.562   0.636    FALSE 1.000 1.000  3360
sa[2]      0.548 0.018   0.512   0.548   0.583    FALSE 1.000 1.001  1948
sa[3]      0.542 0.015   0.512   0.542   0.571    FALSE 1.000 1.001  1269
[...output truncated...]
sa[17]     0.536 0.019   0.498   0.536   0.574    FALSE 1.000 1.001  1401
sa[18]     0.531 0.025   0.482   0.531   0.581    FALSE 1.000 1.001  1678
sa[19]     0.533 0.022   0.490   0.533   0.577    FALSE 1.000 1.001  1554
mean.p     0.603 0.021   0.562   0.602   0.643    FALSE 1.000 1.001  2356
deviance 462.075 3.227 457.799 461.411 470.013    FALSE 1.000 1.001  3362
```

```
# Produce plot for posteriors of beta
par(cex=1.2)
plot(density(out4$sims.list$beta[,2]), type="l", lwd=2, main=NA, axes=FALSE,
xlab="Strength of density dependence")
lines(density(out4$sims.list$beta[,1]), col="blue", lwd=2)
axis(1)
axis(2, las=1)
legend("topright", col=c("blue", "black"), lwd=c(2, 2), legend=c("Juvenile
survival", "Adult survival"), bty="n")
abline(v=0, lty=2)
```



With the specified MCMC settings the model converges fine. The posterior distributions of the density-dependence parameters (beta) do not indicate any evidence for density dependence in survival: both are more or less centered on zero, as we can see in the figure above. (Note: This result was to be expected, since the data are simulated and no density dependence on survival was built into this data set.)

Exercise 4

Use the mark-resighting-recovery data set of the storks (`data(stork)` in the `IPMbook` package). Write a new integrated model that uses as one data source the capture-resighting data, and as another source the dead-recovery data. Each of these data sets has the same number of marked individuals, but one then only includes the resightings and the other only the dead recoveries. True survival is a shared parameter between the two likelihoods in this integrated model. Compare the parameter estimates with those from the joint model developed in Section 4.5.4. Why are the posterior standard deviations different?

Solution. We first create the two data sets, one with the mark-resighting data only, and the other with the dead-recoveries only:

```
# Load data
library(IPMbook)
data(stork)

# Create a data set with the life resightings only
LR <- stork
LR[LR==2] <- 0
```

```
# Create a data set with the dead recoveries only
DR <- stork
f <- getFirst(DR)
DR[DR==1] <- 0
for (i in 1:nrow(DR)){
  DR[i,f[i]] <- 1
}
DR[DR==2] <- 1
```

Next, we produce the m-arrays for the two data sets, i.e., a single-state m-array for the mark-resighting data and a dead-recovery m-array for the other data set.

```
marrLR <- marray(LR)
marrDR <- marrayDead(DR)
```

Then we develop the model. We write it such that it will contain two likelihoods, one for the resighting data and the other for the dead-recovery data. The main parameters in the two likelihoods all have different names, hence there is no link yet between the two likelihoods. The desired link between them is included in that part of the model where we define the priors and the linear models. There, we define the survival parameter from the resighting data (ϕ, apparent survival) to be the product of the survival parameter from the dead-recovery data (*s*, true survival) and site fidelity (*f*), hence $\phi = sf$. We define a model with random year effects for survival (*s*) and fixed time effects for the resighting probabilities (*p*). The other parameters are assumed to be constant over time. We then package the necessary data into a list, write the model, define the MCMC setting and launch JAGS.

```
# Data bundle
jags.data <- list(marrLR=marrLR, relLR=rowSums(marrLR), marrDR=marrDR,
relDR=rowSums(marrDR), nyears=ncol(marrDR))
str(jags.data)

# Model file
cat(file="model5.txt", "
model {
# Priors and linear models
mean.s ~ dunif(0, 1)      # Prior for mean survival
mean.f ~ dunif(0, 1)      # Prior for mean site fidelity
mean.r ~ dunif(0, 1)      # Prior for mean recovery

mu <- logit(mean.s)
sigma ~ dunif(0, 10)      # Prior for temporal variability of survival
tau <- pow(sigma, -2)

for (t in 1:(nyears-1)){      # Loop over time intervals/occasions
  phi[t] <- s[t] * mean.f
  p[t] ~ dunif(0, 1)
  r[t] <- mean.r
  logit.s[t] ~ dnorm(mu, tau)
  s[t] <- ilogit(logit.s[t])
} #t

# Likelihoods
# Define the multinomial likelihood for mark-resighting data
for (t in 1:(nyears-1)){
  marrLR[t,1:nyears] ~ dmulti(piLR[t,], relLR[t])
}
# Define the cell probabilities of the m-array
```

```
for (t in 1:(nyears-1)){
  # Main diagonal
  q[t] <- 1 - p[t]            # Probability of non-recapture
  piLR[t,t] <- phi[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    piLR[t,j] <- prod(phi[t:j]) * prod(q[t:(j-1)]) * p[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    piLR[t,j] <- 0
  } #j
} #t
# Last column: probability of non-recapture
for (t in 1:(nyears-1)){
  piLR[t,nyears] <- 1-sum(piLR[t,1:(nyears-1)])
}

# Define the multinomial likelihood for dead-recovery data
for (t in 1:(nyears-1)){
  marrDR[t,1:nyears] ~ dmulti(piDR[t,], relDR[t])
}
# Define the cell probabilities of the m-array
for (t in 1:(nyears-1)){
  # Main diagonal
  piDR[t,t] <- (1-s[t])*r[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    piDR[t,j] <- prod(s[t:(j-1)])*(1-s[j])*r[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    piDR[t,j] <- 0
  } #j
} #t
# Last column: probability of non-recovery
for (t in 1:(nyears-1)){
  piDR[t,nyears] <- 1-sum(piDR[t,1:(nyears-1)])
} #t
}
")


# Initial values
inits <- function(){list(mean.s=runif(1))}

# Parameters monitored
parameters <- c("mean.s", "mean.f", "sigma", "mean.r", "s", "p")

# MCMC settings
ni <- 50000; nb <- 22000; nc <- 3; nt <- 10; na <- 5000

# Call JAGS from R (ART 3 min), check convergence and summarize posteriors
out5 <- jags(jags.data, inits, parameters, "model5.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out5)             # Not shown
print(out5)

JAGS output for model 'model5.txt', generated by jagsUI.
Estimates based on 3 chains of 50000 iterations,
adaptation = 5000 iterations (sufficient),
burn-in = 22000 iterations and thin rate = 10,
```

```
yielding 8400 total samples from the joint posterior.
MCMC ran in parallel for 3.096 minutes at time 2021-10-13 14:51:42.

             mean    sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.s      0.834 0.063   0.732   0.828   0.977    FALSE 1 1.006   458
mean.f      0.897 0.061   0.763   0.900   0.994    FALSE 1 1.005   559
sigma       0.416 0.259   0.026   0.383   1.021    FALSE 1 1.002  1452
mean.r      0.062 0.067   0.028   0.049   0.205    FALSE 1 1.142   466
s[1]        0.815 0.096   0.599   0.822   0.980    FALSE 1 1.003   770
s[2]        0.816 0.091   0.626   0.819   0.979    FALSE 1 1.003   655
[...output truncated...]
s[14]       0.827 0.073   0.696   0.822   0.979    FALSE 1 1.003   745
s[15]       0.842 0.081   0.671   0.845   0.983    FALSE 1 1.003   938
p[1]        0.526 0.249   0.091   0.521   0.965    FALSE 1 1.000  8400
p[2]        0.497 0.154   0.217   0.490   0.806    FALSE 1 1.000  8400
[...output truncated...]
p[14]       0.699 0.049   0.601   0.700   0.793    FALSE 1 1.000  8400
p[15]       0.658 0.071   0.533   0.653   0.817    FALSE 1 1.000  8400
deviance 490.695 7.680 477.119 490.217 506.854    FALSE 1 1.000  3198
```

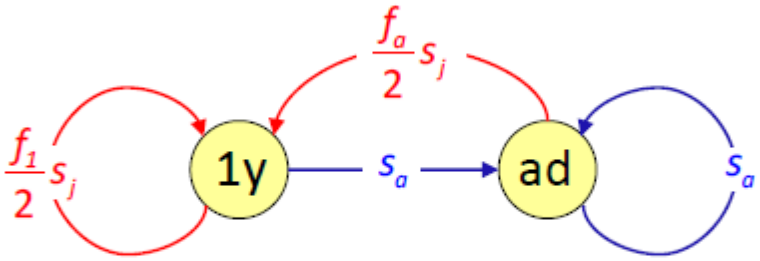
The parameter estimates are similar to those obtained in Section 4.5.4. of the book. The main difference is that now estimates of survival and of the magnitude of temporal variability of survival are both slightly higher than before. Moreover, the precision of the estimates is now reduced compared to that in the joint analysis in Section 4.5.4, (i.e., the posterior standard deviations are greater now). This is unexpected and somehow surprising. We would have expected the opposite, that the precision of the current analysis is (in a spurious manner) improved, because there we commit pseudoreplication here by using the data twice in a single analysis. The mark-resighting data and the dead-recovery data originate from exactly the same individuals; hence the data sets are not independent. Yet, we analyze them as if these individuals were all different, and hence assuming independence of the two data sets. To take the nonindependence into account, the joint model developed in Section 4.5.4 provides the correct solution, while the solution shown here is in fact not correct. However, to be honest, we cannot offer any explanation for why the precision declined in our inappropriate analysis. A simulation study would perhaps help to understand this behavior. It may be that this finding is just due to chance and specific to this data set but does not hold in general.

Chapter 5

Exercise 1

Develop an IPM for the woodchat shrike data that does not include demographic stochasticity and that assumes environmental stochasticity for productivity only, but not for survival. Use independent fixed time effects for productivity of first-time and experienced breeders and fit a prebreeding 'census' model.

Solution: We start by drawing a life-cycle graph and by writing down the population equations. We use a female-based, prebreeding 'census' model, and distinguish between 1-year old ('1y') and older individuals ('ad'). The resulting life-cycle graph is therefore:



We write down the population equations that correspond to this graph:

$$N_{1y,t+1} = s_j \frac{f_{1,t}}{2} N_{1y,t} + s_j \frac{f_{a,t}}{2} N_{ad,t}$$
$$N_{ad,t+1} = s_a \left(N_{1y,t} + N_{ad,t} \right)$$

Next, we load the data, transform the capture-recapture data to the m-array format, bundle them and write the IPM. The code for the IPM should cause no particular difficulties. We here still write the process model with time-dependent demographic rates, but then constrain them according to our wishes, or written in different words, we apply a linear model to make survival constant over time. Written in this more general way the model code has more flexibility in case we want to fit alternative models (e.g., with a logit-linear regression on some covariate). Likewise, the likelihoods of the capture-recapture and of the productivity data need to be written in terms of temporally variable demographic rates.

```
# Load woodchat shrike data
library(IPMbook)
data(woodchat5)

# Transform capture-recapture data into the m-array format
shrike.marray <- marrayAge(woodchat5$ch, woodchat5$age)

# Bundle data
```

```
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=woodchat5$repro[,1],
year=woodchat5$repro[,2], age=woodchat5$repro[,3], C=woodchat5$count)


# Write JAGS model file
cat(file="model1.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  p[t] <- mean.p
}

for (t in 1:n.occasions){
  f[1,t] ~ dunif(0, 10)
  f[2,t] ~ dunif(0, 10)
}

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

sigma.obs ~ dunif(0.5, 100)
tau.obs <- pow(sigma.obs, -2)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: uniform priors
N[1,1] ~ dunif(1, 300)
N[2,1] ~ dunif(1, 300)

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

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

# Productivity data (Poisson regression model)
for (i in 1:length(J)){
  J[i] ~ dpois(f[age[i], year[i]])
}

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

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

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

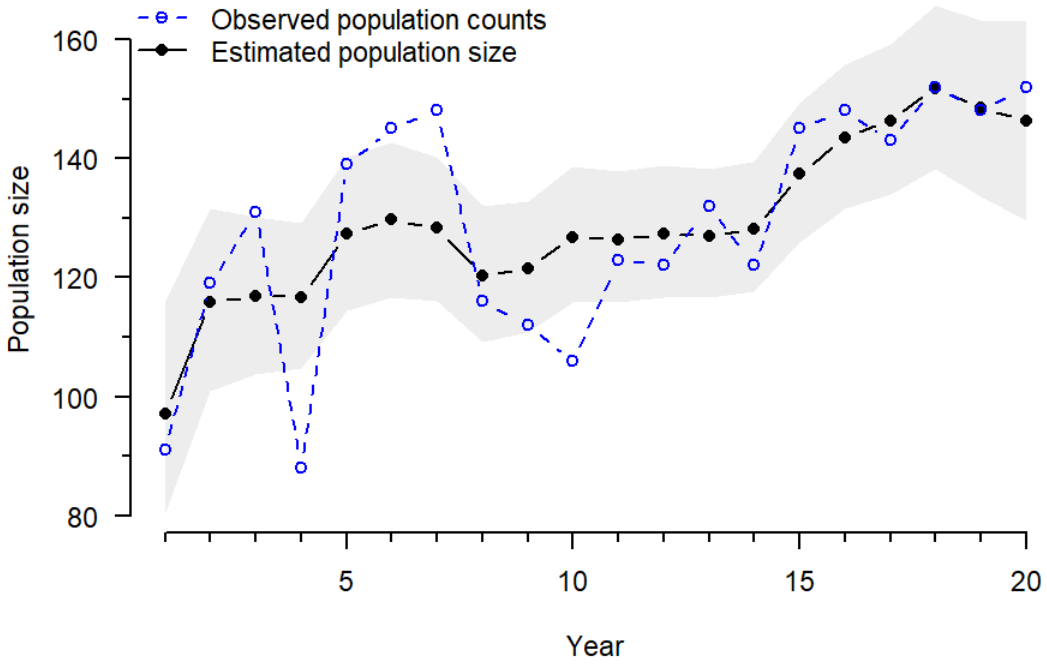
We define initial values, a list of parameters to be monitored, the MCMC settings and launch the model.

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

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

# MCMC settings
ni <- 40000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS (ART 2 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)     # Not shown
print(out1, 3)
```

```
JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 40000 iterations,
adaptation = 3000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 3,
yielding 30000 total samples from the joint posterior.
MCMC ran in parallel for 1.829 minutes at time 2021-10-13 15:08:31.

                       mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
mean.sj               0.299  0.010    0.280    0.299    0.318    FALSE 1 1.004   660
mean.sa               0.540  0.012    0.516    0.540    0.564    FALSE 1 1.004   666
f[1,1]                3.214  0.429    2.419    3.200    4.106    FALSE 1 1.000 10250
f[2,1]                5.509  0.659    4.292    5.477    6.891    FALSE 1 1.000 30000
f[1,2]                2.599  0.399    1.876    2.581    3.430    FALSE 1 1.000 15678
f[2,2]                3.811  0.470    2.943    3.792    4.788    FALSE 1 1.001  3243
[...output truncated...]
f[1,19]               2.306  0.284    1.784    2.297    2.894    FALSE 1 1.000 30000
f[2,19]               3.532  0.394    2.798    3.521    4.338    FALSE 1 1.000 30000
f[1,20]               2.858  0.362    2.200    2.842    3.610    FALSE 1 1.000 14833
```

```
f[2,20]                4.119  0.354    3.455    4.106    4.843    FALSE 1 1.000 16161
mean.p                 0.607  0.019    0.570    0.607    0.644    FALSE 1 1.000 12474
N[1,1]                47.292 29.691    3.346   43.634  104.335    FALSE 1 1.006   398
N[2,1]                49.872 24.759    4.227   52.013   90.511    FALSE 1 1.007   354
N[1,2]                63.489  7.395   49.451   63.483   78.101    FALSE 1 1.002   924
N[2,2]                52.426  4.996   43.286   52.209   62.856    FALSE 1 1.000 10911
[...output truncated...]
N[1,19]               66.515  5.683   55.665   66.437   77.983    FALSE 1 1.001  3642
N[2,19]               81.913  4.241   73.728   81.884   90.447    FALSE 1 1.001  3948
N[1,20]               66.148  6.131   54.473   66.047   78.498    FALSE 1 1.001  3949
N[2,20]               80.096  4.556   71.144   80.052   89.085    FALSE 1 1.000  6535
sigma.obs             13.941  2.910    9.383   13.564   20.718    FALSE 1 1.000 30000
ann.growth.rate[1]     1.200  0.107    1.006    1.197    1.421    FALSE 1 1.003   690
ann.growth.rate[2]     1.010  0.046    0.925    1.009    1.104    FALSE 1 1.001  2042
[...output truncated...]
ann.growth.rate[18]    0.978  0.036    0.910    0.977    1.051    FALSE 1 1.000 30000
ann.growth.rate[19]    0.986  0.037    0.915    0.985    1.061    FALSE 1 1.000 30000
Ntot[1]               97.164  9.035   80.563   96.813  116.092    FALSE 1 1.001  2667
Ntot[2]              115.915  7.739  100.919  115.808  131.472    FALSE 1 1.001  1486
[...output truncated...]
Ntot[19]             148.428  7.505  133.499  148.414  163.182    FALSE 1 1.000 30000
Ntot[20]             146.244  8.510  129.459  146.301  162.944    FALSE 1 1.000 29793
deviance            4291.977  9.772 4274.886 4291.338 4313.171    FALSE 1 1.000 26353
```

```
# Code for a graph of total population size
par(mar=c(4, 4, 3, 0), las=1, cex=1.4, lwd=1.5)
u <- col2rgb("grey82")
T <- length(woodchat5$count)
col.pol <- rgb(u[1], u[2], u[3], alpha=100, maxColorValue=255)
plot(out1$mean$Ntot, type="n", ylim=range(c(out1$q2.5$Ntot, out1$q97.5$Ntot,
woodchat5$count)), ylab="Population size", xlab="Year", las=1, cex=1.5,
axes=FALSE)
axis(2, las=1, lwd=lwd)
axis(2, at=c(90, 110, 130, 150), labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=1:T, labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5, lwd=lwd)
polygon(c(1:T, T:1), c(out1$q2.5$Ntot, out1$q97.5$Ntot[T:1]), border=NA,
col=col.pol)
points(out1$mean$Ntot, type="b", col="black", pch=16, lty=1, lwd=lwd)
points(woodchat5$count, type="b", col="blue", pch=1, lty=2, lwd=lwd)
legend("topleft", legend=c("Observed population counts", "Estimated population
size"), pch=c(1, 16), lwd=c(lwd, lwd), col=c("blue", "black"), lty=c(2, 1),
bty="n")
```

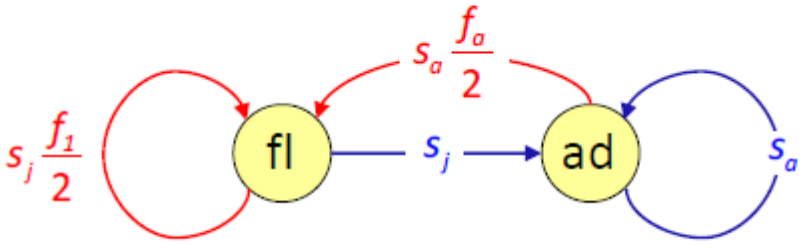


The estimated trajectory of total population size is very similar to that obtained under the IPM that includes environmental stochasticity for all demographic parameters (Fig. 5.3 in the book). The reason for why there is not much difference is the lack of temporal variability in the demographic rates – recall that these are simulated data and that these were simulated under the assumption of constant demographic rates.

Exercise 2

Develop an IPM for the woodchat shrike data with the same features as in Exercise 1, but now adopt a postbreeding 'census' population model.

Solution: As always, we start by drawing a life-cycle graph and then writing down the corresponding population equations. The stage classes that we define for the postbreeding 'census' model are now: 'fledglings' and individuals that are at least 1-year old, i.e., the 'adults'. Here is the life-cycle:



We write down the population equations corresponding to that graph:

$$N_{fl,t+1} = s_j \frac{f_{1,t+1}}{2} N_{fl,t} + s_a \frac{f_{a,t+1}}{2} N_{ad,t}$$
$$N_{ad,t+1} = s_j N_{fl,t} + s_a N_{ad,t}$$

We then again load the data, adapt the model from the previous exercise so that it fits to the postbreeding census, and then run the model. Compared to the prebreeding 'census' model in Exercise 1, some changes are necessary: *i*) Only breeders are included in the population count, which are now the individuals of the second stage class. Hence, the observation model needs to be restricted to the second stage class. *ii*) We need a prior for the initial population size of the fledglings. *iii*) And of course, the population model must be changed such that it matches the model developed above. iv) The annual population growth rate is calculated based on the number of breeding individuals, i.e., excluding the fledglings.

```
# Load woodchat shrike data
library(IPMbook)
data(woodchat5)

# Transform capture-recapture data into the m-array format
shrike.marray <- marrayAge(woodchat5$ch, woodchat5$age)

# Bundle data
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=woodchat5$repro[,1],
year=woodchat5$repro[,2], age=woodchat5$repro[,3], C=woodchat5$count)


# Write JAGS model file
cat(file="model2.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  p[t] <- mean.p
}

for (t in 1:n.occasions){
```

```
  f[1,t] ~ dunif(0, 10)
  f[2,t] ~ dunif(0, 10)
}

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

sigma.obs ~ dunif(0.5, 100)
tau.obs <- pow(sigma.obs, -2)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: uniform priors
N[1,1] ~ dunif(1, 1000)   # larger than for pre-breding census
N[2,1] ~ dunif(1, 300)

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

# Observation model
for (t in 1:n.occasions){
  C[t] ~ dnorm(N[2,t], tau.obs)   # fledglings are not included in the counts,
hence we include only breeders (2nd stage class)
}

# Productivity data (Poisson regression model)
for (i in 1:length(J)){
  J[i] ~ dpois(f[age[i], year[i]])
}

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

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

We define initial values, the parameters to be monitored, the MCMC settings and unleash JAGS to experience the raw power of its MCMC algorithms.

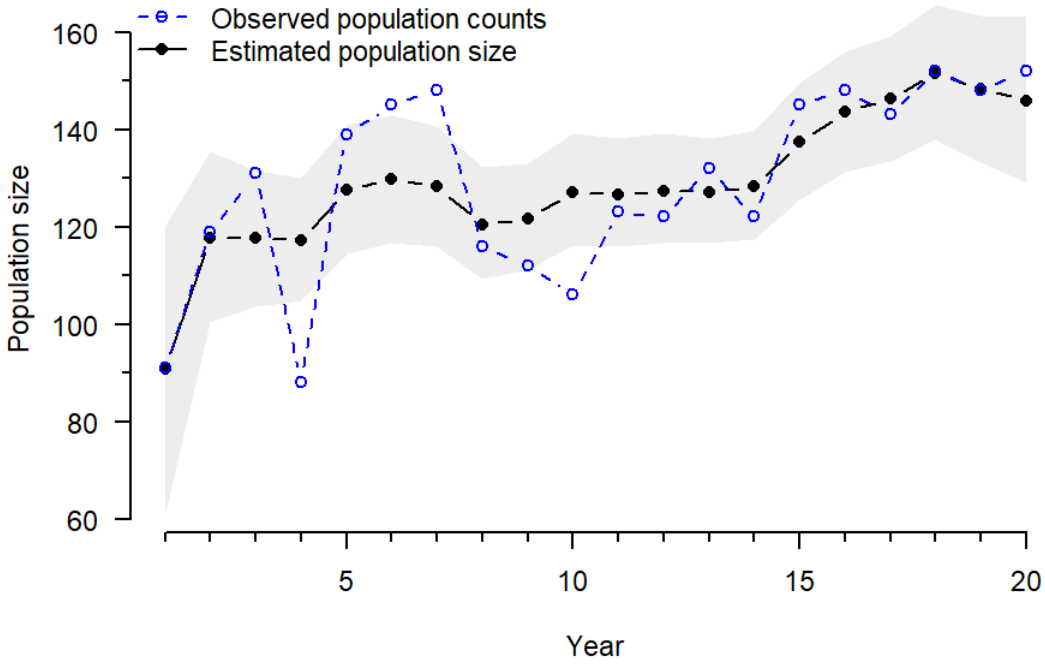
```
# Initial values
inits <- function(){list(mean.sj=runif(1, 0, 0.5))}

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

# MCMC settings
ni <- 40000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS (ART 2 min), check convergence and summarize posteriors
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out2)     # Not shown
print(out2, 3)
```

```
JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 3 chains of 40000 iterations,
adaptation = 3000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 3,
yielding 30000 total samples from the joint posterior.
MCMC ran in parallel for 1.753 minutes at time 2021-10-13 15:17:37.

                         mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
mean.sj                 0.298  0.010    0.280    0.298    0.317    FALSE 1 1.003  1294
mean.sa                 0.540  0.012    0.516    0.540    0.563    FALSE 1 1.004   928
f[1,1]                  3.178  0.433    2.383    3.160    4.069    FALSE 1 1.000 30000
f[2,1]                  5.500  0.678    4.244    5.471    6.910    FALSE 1 1.000 10165
f[1,2]                  2.577  0.395    1.858    2.559    3.401    FALSE 1 1.000 18922
f[2,2]                  3.803  0.475    2.914    3.788    4.776    FALSE 1 1.000 14577
[...output truncated...]
f[1,19]                 2.306  0.283    1.782    2.296    2.890    FALSE 1 1.000 12396
f[2,19]                 3.531  0.397    2.788    3.519    4.345    FALSE 1 1.000 29764
f[1,20]                 2.869  0.363    2.209    2.850    3.617    FALSE 1 1.000 30000
f[2,20]                 4.120  0.354    3.455    4.109    4.849    FALSE 1 1.000 30000
mean.p                  0.607  0.019    0.570    0.607    0.644    FALSE 1 1.000 23333
N[1,1]                230.568 43.200  143.679  230.666  315.781    FALSE 1 1.001  6130
N[2,1]                 90.829 14.731   61.350   90.766  120.215    FALSE 1 1.001  9109
N[1,2]                181.225 16.355  149.907  180.960  214.308    FALSE 1 1.000 30000
N[2,2]                117.746  8.867  100.311  117.713  135.337    FALSE 1 1.000  7995
[...output truncated...]
N[1,19]               220.896 19.375  184.124  220.609  259.866    FALSE 1 1.000 30000
N[2,19]               148.232  7.623  133.181  148.213  163.266    FALSE 1 1.000 30000
N[1,20]               259.420 23.573  215.556  258.596  308.604    FALSE 1 1.000 30000
N[2,20]               145.935  8.621  128.948  145.924  162.996    FALSE 1 1.000 30000
sigma.obs              14.239  3.069    9.555   13.804   21.443    FALSE 1 1.000 17969
ann.growth.rate[1]      1.336  0.278    0.919    1.299    1.979    FALSE 1 1.001  4125
ann.growth.rate[2]      1.001  0.049    0.909    0.999    1.101    FALSE 1 1.000 19559
[...output truncated...]
ann.growth.rate[18]     0.978  0.036    0.910    0.977    1.050    FALSE 1 1.000 30000
ann.growth.rate[19]     0.985  0.038    0.914    0.984    1.061    FALSE 1 1.000 30000
```

```
Ntot[1]              321.397 34.773  253.101  321.365  390.432    FALSE 1 1.000  7655
Ntot[2]              298.972 20.121  259.853  298.801  339.141    FALSE 1 1.000 30000
[...output truncated...]
Ntot[19]             369.129 23.265  323.520  368.999  415.535    FALSE 1 1.000 30000
Ntot[20]             405.355 29.652  348.849  404.519  466.372    FALSE 1 1.000 30000
deviance            4292.747  9.944 4275.102 4292.154 4314.026    FALSE 1 1.000 20221
```

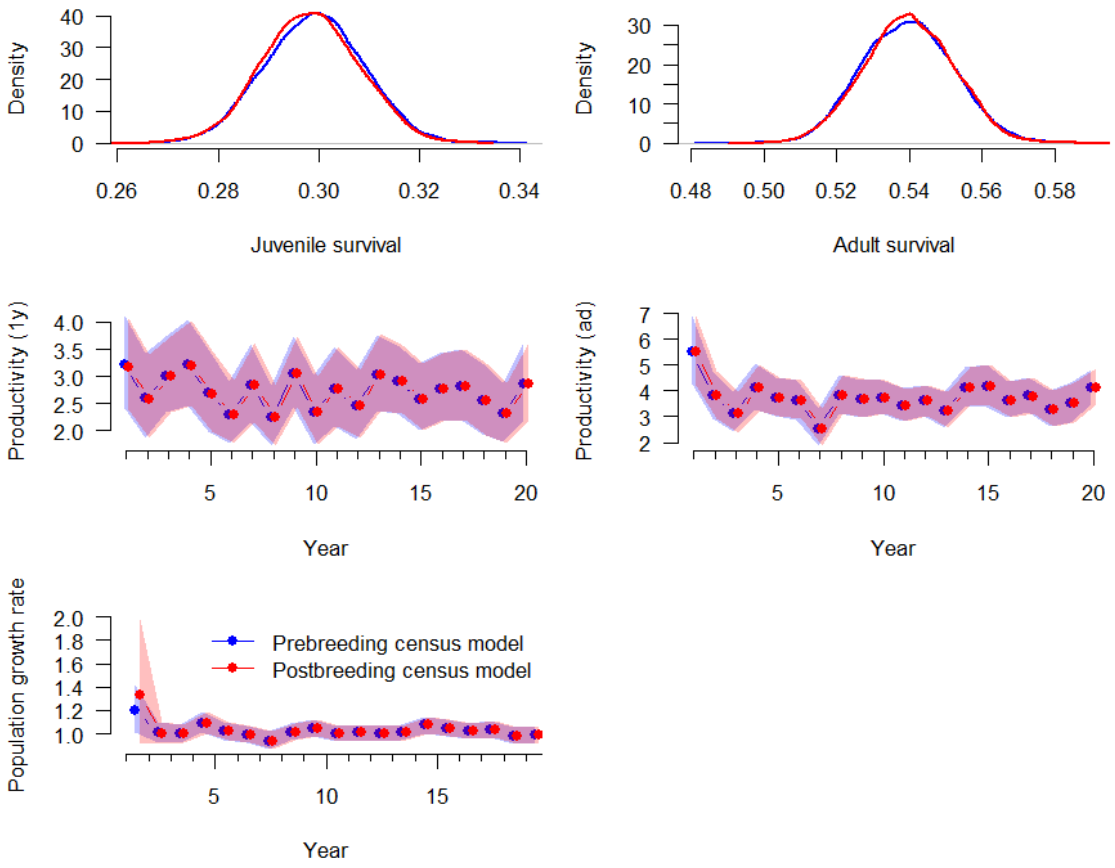
We now also produce a plot that shows the estimated vs. counted number of breeders.

```
# Code for a graph of the number of breeders
par(mar=c(4, 4, 3, 0), las=1, cex=1.4, lwd=1.5)
u <- col2rgb("grey82")
T <- length(woodchat5$count)
col.pol <- rgb(u[1], u[2], u[3], alpha=100, maxColorValue=255)
plot(out2$mean$N[2,], type="n", ylim=range(c(out2$q2.5$N[2,], out2$q97.5$N[2,],
woodchat5$count)), ylab="Population size", xlab="Year", las=1, cex=1.5,
axes=FALSE)
axis(2, las=1, lwd=lwd)
axis(2, at=c(90, 110, 130, 150), labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=1:T, labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5, lwd=lwd)
polygon(c(1:T, T:1), c(out2$q2.5$N[2,], out2$q97.5$N[2,T:1]), border=NA,
col=col.pol)
points(out2$mean$N[2,], type="b", col="black", pch=16, lty=1, lwd=lwd)
points(woodchat5$count, type="b", col="blue", pch=1, lty=2, lwd=lwd)
legend("topleft", legend=c("Observed population counts", "Estimated population
size"), pch=c(1, 16), lwd=c(lwd, lwd), col=c("blue", "black"), lty=c(2, 1),
bty="n")
```



Next, we compare the estimates of the demographic parameters and the population growth rate between the models that use a pre- and a postbreeding 'census'.

```
par(mfrow=c(3, 2), mar=c(4, 4, 2, 1), las=1, cex=1.1)
T <- length(woodchat5$count)
plot(density(out1$sims.list$mean.sj), col="blue", xlab="Juvenile survival",
main=NA, axes=FALSE, lwd=2)
lines(density(out2$sims.list$mean.sj), col="red", lwd=2)
axis(1); axis(2, las=1)

plot(density(out1$sims.list$mean.sa), col="blue", xlab="Adult survival",
main=NA, axes=FALSE, lwd=2)
lines(density(out2$sims.list$mean.sa), col="red", lwd=2)
axis(1); axis(2, las=1)

library(scales)
d <- 0.1
plot(out1$mean$f[1,], type="n", ylim=range(c(out1$q2.5$f[1,],
out1$q97.5$f[1,])), ylab="Productivity (1y)", xlab="Year", las=1, cex=1.5,
axes=FALSE)
axis(2, las=1)
axis(1, at=1:T, labels=NA, tcl=-0.25)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5)
polygon(c((1:T)-d, (T:1)-d), c(out1$q2.5$f[1,], rev(out1$q97.5$f[1,])),
border=NA, col=alpha("blue", 0.25))
points(y=out1$mean$f[1,], x=(1:T)-d, type="b", col="blue", pch=16, lty=1)
polygon(c((1:T)+d, (T:1)+d), c(out2$q2.5$f[1,], rev(out2$q97.5$f[1,])),
border=NA, col=alpha("red", 0.25))
points(y=out2$mean$f[1,], x=(1:T)+d, type="b", col="red", pch=16, lty=1)

plot(out1$mean$f[2,], type="n", ylim=range(c(out1$q2.5$f[2,],
out1$q97.5$f[2,])), ylab="Productivity (ad)", xlab="Year", las=1, cex=1.5,
axes=FALSE)
axis(2, las=1)
axis(1, at=1:T, labels=NA, tcl=-0.25)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5)
polygon(c((1:T)-d, (T:1)-d), c(out1$q2.5$f[2,], rev(out1$q97.5$f[2,])),
border=NA, col=alpha("blue", 0.25))
points(y=out1$mean$f[2,], x=(1:T)-d, type="b", col="blue", pch=16, lty=1)
polygon(c((1:T)+d, (T:1)+d), c(out2$q2.5$f[2,], rev(out2$q97.5$f[2,])),
border=NA, col=alpha("red", 0.25))
points(y=out2$mean$f[2,], x=(1:T)+d, type="b", col="red", pch=16, lty=1)

plot(out1$mean$ann.growth.rate, type="n",
ylim=range(c(out2$q2.5$ann.growth.rate, out2$q97.5$ann.growth.rate)),
ylab="Population growth rate", xlab="Year", las=1, cex=1.5, axes=FALSE)
axis(2, las=1)
axis(1, at=1:T, labels=NA, tcl=-0.25)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5)
polygon(c((1:(T-1))+0.5-d, ((T-1):1)+0.5-d), c(out1$q2.5$ann.growth.rate,
rev(out1$q97.5$ann.growth.rate)), border=NA, col=alpha("blue", 0.25))
points(y=out1$mean$ann.growth.rate, x=(1:(T-1))+0.5-d, type="b", col="blue",
pch=16, lty=1)
polygon(c((1:(T-1))+0.5+d, ((T-1):1)+0.5+d), c(out2$q2.5$ann.growth.rate,
rev(out2$q97.5$ann.growth.rate)), border=NA, col=alpha("red", 0.25))
points(y=out2$mean$ann.growth.rate, x=(1:(T-1))+0.5+d, type="b", col="red",
pch=16, lty=1)
legend("topright", legend=c("Prebreeding census model", "Postbreeding census
model"), pch=rep(16, 2), lwd=rep(1, 2), col=c("blue", "red"), bty="n")
```



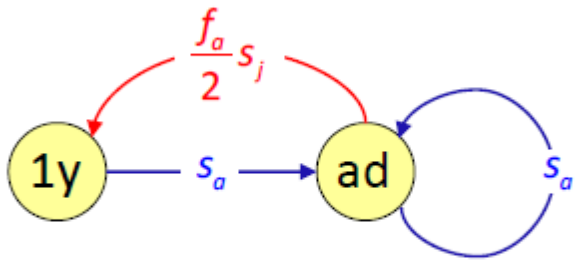
We clearly see that under both models demographic rates, age-dependent survival (posterior distributions) and productivity (annual posterior means with 95% credible intervals) are essentially identical. Hence, the estimates of these parameters were not affected by our choice of a pre- or postbreeding 'census' population model. The same is true for the annual population growth rates. We here see only a small difference for the population growth rates between the first and the second year. This has to do with the estimation of the population sizes in the first year, which is difficult, as there is no Markovian structure that can help with the estimation of the initial *N*. Thus, we advise care when interpreting estimates of initial population sizes and quantities that are derived from them.

Exercise 3

Develop an IPM for the woodchat shrike data sets, but now assume erroneously that 1-year old individuals do not yet reproduce. You may also include demographic stochasticity (but don't add temporal variability, i.e., environmental stochasticity). Compare the counts and the estimated population sizes and also compare the residual error with that obtained from Exercise 1. What do you observe and how can you interpret your observations?

Solution: We start again by drawing a life-cycle graph and by writing down the population equations. This is now particularly important, since we need to change the population model compared to the models entertained in Exercise 1 and 2 or at other places throughout Chapter 5.

We develop a prebreeding 'census' model. As a consequence, there is one important change in the life-cycle graph compared to that in Exercise 1: as the shrikes are assumed not to reproduce at age one, the self-loop of the node "1y" now vanishes. All the other parts of the life-cycle graph remain the same:



The stage-specific population equations need to be adapted accordingly. We also include demographic stochasticity and therefore get:

$$N_{1y,t+1} \sim Poisson\left(s_j \frac{f_a}{2} N_{ad,t} \right)$$

$$N_{ad,t+1} \sim Binomial\left(s_a, N_{1y,t} + N_{ad,t} \right)$$

Next, we load and bundle the data. Since we assume no reproduction at the age of 1-year, we consequently do not have productivity data from this age class.

```
# Load woodchat shrike data
library(IPMbook)
data(woodchat5)

# Transform capture-recapture data into the m-array format
shrike.marray <- marrayAge(woodchat5$ch, woodchat5$age)

# Bundle data
jags.data <- with(woodchat5, list(marr.j=shrike.marray[,,1],
marr.a=shrike.marray[,,2], n.occasions=dim(shrike.marray)[2],
rel.j=rowSums(shrike.marray[,,1]), rel.a=rowSums(shrike.marray[,,2]),
J=repro[repro[,3]==2,1], year=repro[repro[,3]==2,2], C=count, pNinit=dUnif(1,
300)))


# Write JAGS model file
cat(file="model3.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  p[t] <- mean.p
}

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

```
}

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.f ~ dunif(0, 10)
mean.p ~ dunif(0, 1)

sigma.obs ~ dunif(0.5, 100)
tau.obs <- pow(sigma.obs, -2)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

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

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

# Productivity data (Poisson regression model)
for (i in 1:length(J)){
  J[i] ~ dpois(f[year[i]])
}

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

# Derived parameters
# Annual population growth rate (added 0.001 to avoid possible division by 0)
```

```
for (t in 1:(n.occasions-1)){
  ann.growth.rate[t] <- (N[1,t+1] + N[2,t+1]) / (N[1,t] + N[2,t] + 0.001)
}
# Total population size
for (t in 1:n.occasions){
  Ntot[t] <- N[1,t] + N[2,t]
}
}
")
```

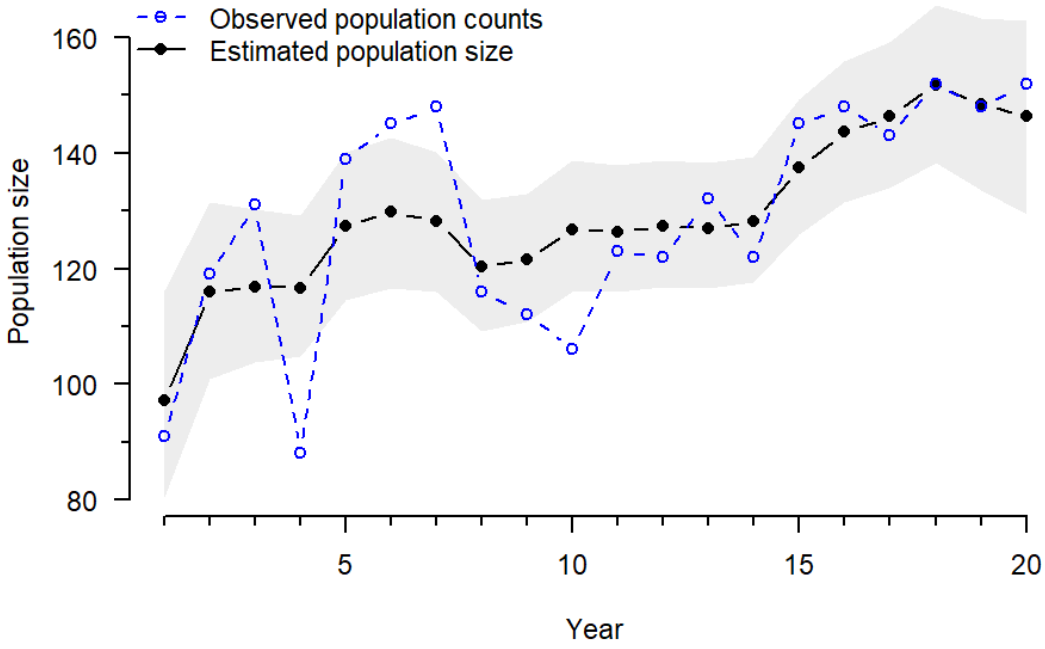
We define the list with the initial values, define the parameters to be monitored, choose the MCMC settings and run the model.

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

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

# MCMC settings
ni <- 40000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS (ART 2 min), check convergence and summarize posteriors
out3 <- jags(jags.data, inits, parameters, "model3.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out3)     # Not shown
print(out3, 3)
```

```
JAGS output for model 'model3.txt', generated by jagsUI.
Estimates based on 3 chains of 40000 iterations,
adaptation = 3000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 3,
yielding 30000 total samples from the joint posterior.
MCMC ran in parallel for 1.756 minutes at time 2021-10-13 15:47:11.

                       mean     sd      2.5%       50%    97.5% overlap0 f  Rhat n.eff
mean.sj               0.345  0.015     0.316     0.345    0.376    FALSE 1 1.000 14568
mean.sa               0.582  0.012     0.559     0.582    0.604    FALSE 1 1.000 30000
mean.f                3.785  0.086     3.615     3.784    3.955    FALSE 1 1.000 16963
mean.p                0.569  0.020     0.529     0.569    0.609    FALSE 1 1.000 23905
N[1,1]               28.380 22.178     2.000    23.000   84.000    FALSE 1 1.000 30000
N[2,1]               81.599 17.451    42.000    84.000  111.000    FALSE 1 1.000 16751
N[1,2]               54.650 11.685    29.000    56.000   75.000    FALSE 1 1.000 28328
N[2,2]               66.533  7.864    53.000    66.000   84.000    FALSE 1 1.001  6393
[...output truncated...]
N[1,19]              57.751  7.670    43.000    58.000   73.000    FALSE 1 1.000  9347
N[2,19]              86.246  7.035    72.000    86.000  100.000    FALSE 1 1.000  7258
N[1,20]              58.218  7.969    43.000    58.000   74.000    FALSE 1 1.000 24578
N[2,20]              85.045  7.773    69.000    85.000  100.000    FALSE 1 1.000  6134
sigma.obs            15.285  4.070     9.110    14.673   24.832    FALSE 1 1.000 26860
ann.growth.rate[1]    1.112  0.128     0.839     1.119    1.344    FALSE 1 1.000 30000
ann.growth.rate[2]    0.997  0.083     0.850     0.992    1.176    FALSE 1 1.000 15145
[...output truncated...]
ann.growth.rate[18]   0.996  0.062     0.879     0.993    1.121    FALSE 1 1.000 29175
ann.growth.rate[19]   0.996  0.064     0.872     0.993    1.126    FALSE 1 1.000 30000
Ntot[1]             109.979 12.345    89.000   109.000  138.000    FALSE 1 1.001  3891
Ntot[2]             121.183 10.695   101.000   121.000  143.000    FALSE 1 1.000  4414
[...output truncated...]
Ntot[19]            143.997  9.880   123.000   144.000  163.000    FALSE 1 1.000  4361
Ntot[20]            143.263 11.319   119.000   144.000  164.000    FALSE 1 1.000  6452
deviance           2639.744  8.743 2622.847 2639.646 2657.149    FALSE 1 1.000 13786
```

```
# Code for a graph of the number of breeders
par(mar=c(4, 4, 3, 0), las=1, cex=1.4, lwd=1.5)
u <- col2rgb("grey82")
T <- length(woodchat5$count)
```

```
col.pol <- rgb(u[1], u[2], u[3], alpha=100, maxColorValue=255)
plot(out1$mean$Ntot, type="n", ylim=range(c(out1$q2.5$Ntot, out1$q97.5$Ntot,
woodchat5$count)), ylab="Population size", xlab="Year", las=1, cex=1.5,
axes=FALSE)
axis(2, las=1, lwd=lwd)
axis(2, at=c(90, 110, 130, 150), labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=1:T, labels=NA, tcl=-0.25, lwd=lwd)
axis(1, at=c(5, 10, 15, 20), labels=c(5, 10, 15, 20), tcl=-0.5, lwd=lwd)
polygon(c(1:T, T:1), c(out1$q2.5$Ntot, out1$q97.5$Ntot[T:1]), border=NA,
col=col.pol)
points(out1$mean$Ntot, type="b", col="black", pch=16, lty=1, lwd=lwd)
points(woodchat5$count, type="b", col="blue", pch=1, lty=2, lwd=lwd)
legend("topleft", legend=c("Observed population counts", "Estimated population
size"), pch=c(1, 16), lwd=c(lwd, lwd), col=c("blue", "black"), lty=c(2, 1),
bty="n")
```

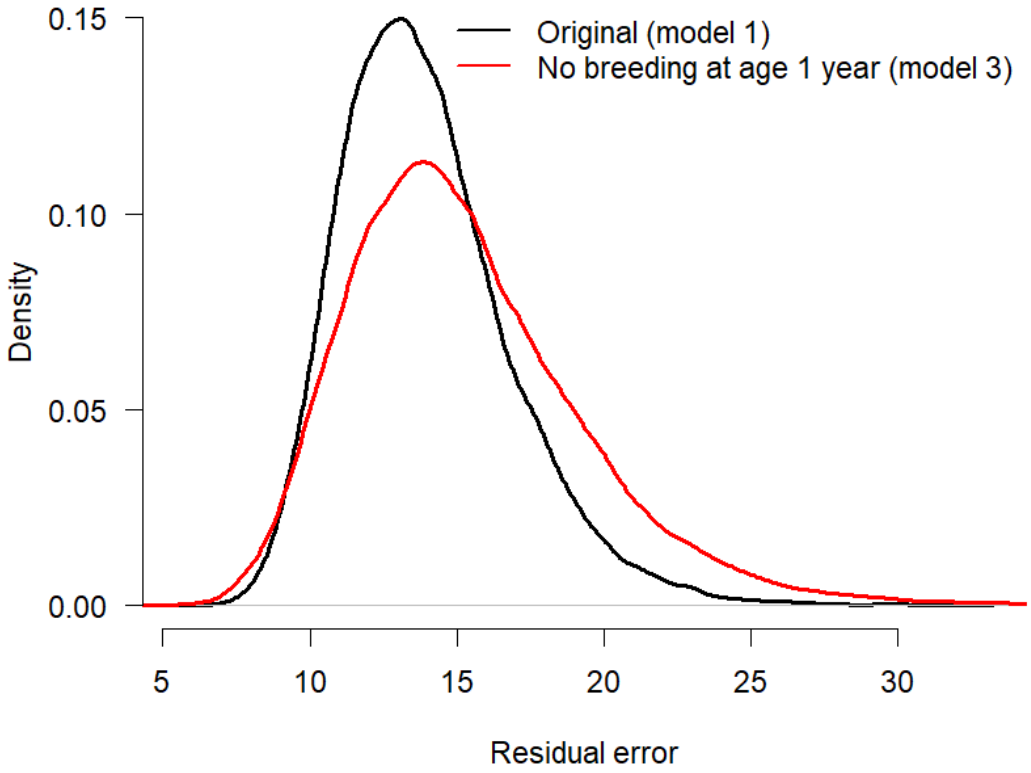


Although a wrong population model was fit, there is quite a good match between the estimated total population sizes and the counts. The good agreement is possible because the estimates of the demographic parameters are different compared to those under the correct model 1. For example, the posterior mean of juvenile survival is 0.345 under the current model, while it was 0.299 under model 1 (remember that your estimates will differ slightly due to MC error). For adult survival, the difference is less, but still there (now: 0.582; model 1: 0.540). See Section 7.2.1 for more explanation of this behavior of IPMs.

We think that this is an important observation which shows once more that a good agreement between the estimated latent states *N* and the observed counts offers no guarantee that the model really "fits". Although we have a demonstrably ill-fitting model here, the match between the two time series is pretty good.

The residual error of the IPM is expected to become larger when the process model is wrong. We can demonstrate that by comparing the posterior distributions of the residual errors under the current and under model 1.

```
par(lwd=3, mar=c(4, 4, 2, 1))
plot(density(out1$sims.list$sigma.obs), xlab="Residual error", main=NA,
axes=FALSE)
lines(density(out3$sims.list$sigma.obs), col="red")
axis(1); axis(2, las=1)
legend("topright", lwd=c(2,2), col=c("black", "red"), legend=c("Original (model
1)", "No breeding at age 1 year (model 3)"), bty="n")
```



We see the behavior we expect: the residual error is inflated when the wrong model is fitted. However, the difference is quite small here. When the sample sizes are increased, the differences between the demographic rates under the two models are expected to decline and the difference between the two estimates of the residual error to increase.

Chapter 6

Exercise 1

Investigate the gain in precision when access to structured counts is available compared to when you have simple scalar counts without stage structure. Explore these questions in the context of the woodchat shrike data set, assuming you have capture-recapture data and structured counts, i.e., vectors of counts of 1-year old and 2-year-and-older individuals. For comparison, also fit a model that uses the unstructured counts, i.e., the sum of the structured counts. Construct a model with constant demographic rates across time but accommodating demographic stochasticity. Use the data provided on solution file.

Here are the data that we need for this exercise:

```
marr.j <- matrix(c(23,8,1,0,0,0,0,0,0,204,
0,29,6,0,0,0,0,0,0,129,
0,0,14,2,0,1,0,0,0,125,
0,0,0,20,7,1,1,0,0,73,
0,0,0,0,15,3,1,0,0,113,
0,0,0,0,0,21,8,2,0,94,
0,0,0,0,0,0,29,2,0,107,
0,0,0,0,0,0,0,7,3,115,
0,0,0,0,0,0,0,0,11,91), ncol=10, nrow=9, byrow=TRUE)

marr.a <- matrix(c(34,5,1,0,0,0,0,0,0,81,
0,32,4,4,1,0,0,0,0,45,
0,0,18,8,2,1,0,0,0,57,
0,0,0,17,2,1,0,0,0,24,
0,0,0,0,15,5,2,0,0,33,
0,0,0,0,0,16,1,0,0,28,
0,0,0,0,0,0,28,3,0,25,
0,0,0,0,0,0,0,21,3,58,
0,0,0,0,0,0,0,0,10,33), ncol=10, nrow=9, byrow=TRUE)

count.1 <- c(88,64,53,8,35,38,58,64,38,28)
count.2 <- c(104,82,82,70,23,36,51,45,66,22)
```

Solution: The life-cycle graph and population model are identical to what you saw in Section 3.2.3, and the IPM model code is similar to that seen in Section 5.4.4. The main difference is that now there are two observation models for the counts, one for each stage class. We assume that the residual errors are identical for both, but we could also adopt different observation variances for the counts of each stage class. Besides capture-recapture and productivity data, we now have also structured population counts. We load these data, bundle them into a list and write the model.

```
# Bundle data
jags.data <- list(marr.j=marr.j, marr.a=marr.a, n.occasions=dim(marr.j)[2],
rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), C1=count.1, C2=count.2,
pNinit=dUnif(1, 300))


# Specify the model in BUGS language
cat(file="model1.txt", "
model {
# Priors and linear models
```

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

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma ~ dunif(0.5, 100)
tau <- pow(sigma, -2)

# State-space model for population count data
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

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

# Observation model
for (t in 1:n.occasions){
  C1[t] ~ dnorm(N[1,t], tau)
  C2[t] ~ dnorm(N[2,t], tau)
}

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

# Derived parameters
```

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

Next, we define the lists with initial values and with the parameters to be monitored, choose the MCMC settings and run the model.

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

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

# MCMC settings
ni <- 40000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS (ART 1 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)    # Not shown
print(out1, 3)


JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 40000 iterations,
adaptation = 3000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 3,
yielding 30000 total samples from the joint posterior.
MCMC ran in parallel for 0.894 minutes at time 2021-10-13 19:58:12.

             mean      sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj     0.235   0.016   0.205   0.235   0.269    FALSE 1 1.000 19779
mean.sa     0.520   0.018   0.484   0.520   0.556    FALSE 1 1.000 30000
mean.f      3.215   0.406   2.447   3.204   4.048    FALSE 1 1.000  8511
mean.p      0.578   0.029   0.522   0.578   0.634    FALSE 1 1.000  8289
N[1,1]     76.360  17.291  42.000  76.000 111.000    FALSE 1 1.000 30000
N[2,1]     92.249  17.408  57.000  92.000 127.000    FALSE 1 1.000 27877
N[1,2]     61.134   8.683  44.000  61.000  79.000    FALSE 1 1.000  5506
N[2,2]     85.477  10.920  64.000  85.000 107.000    FALSE 1 1.000 18455
[...output truncated...]
N[1,9]     30.077   8.208  15.000  30.000  47.000    FALSE 1 1.001  2468
N[2,9]     42.439   8.177  26.000  42.000  58.000    FALSE 1 1.000  4694
N[1,10]    27.471   8.558  12.000  27.000  46.000    FALSE 1 1.000  5909
N[2,10]    36.795   8.323  21.000  37.000  54.000    FALSE 1 1.001  2557
sigma      19.312   4.211  12.738  18.751  29.108    FALSE 1 1.000 22735
Ntot[1]   168.610  20.664 128.000 169.000 210.000    FALSE 1 1.000 30000
Ntot[2]   146.611  15.238 116.000 147.000 177.000    FALSE 1 1.001  6127
[...output truncated...]
Ntot[9]    72.516  14.522  44.000  72.000 101.000    FALSE 1 1.001  2655
Ntot[10]   64.266  15.275  35.000  64.000  96.000    FALSE 1 1.001  3160
deviance 429.131   5.829 418.383 428.857 441.531    FALSE 1 1.000 30000
```

Next, we write a further model that uses the unstructured population counts. Data of unstructured counts are obtained by summing the counts over the two stage classes.

```
# Bundle data
C <- count.1 + count.2
```

```
jags.data <- list(marr.j=marr.j, marr.a=marr.a, n.occasions=dim(marr.j)[2],
rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), C=C, pNinit=dUnif(1, 300))


# Specify the model in BUGS language
cat(file="model2.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  p[t] <- mean.p
  f[t] <- mean.f
}

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma ~ dunif(0.5, 100)
tau <- pow(sigma, -2)

# State-space model for population count data
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

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

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

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

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

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

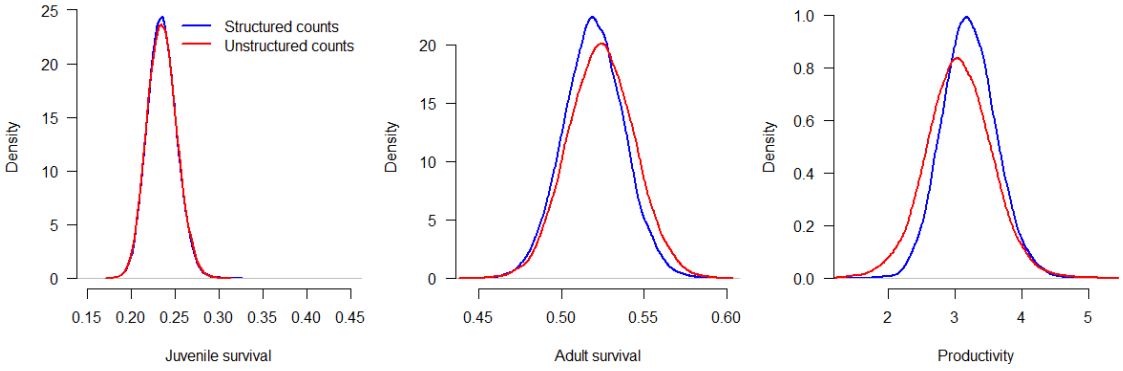
We complete the other preparations and run the model.

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

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

# MCMC settings
ni <- 40000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS (ART 1 min), check convergence and summarize posteriors
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out2)     # Not shown
print(out2, 3)
```

```
JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 3 chains of 40000 iterations,
adaptation = 3000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 3,
yielding 30000 total samples from the joint posterior.
MCMC ran in parallel for 0.908 minutes at time 2021-10-13 20:00:50.

             mean     sd    2.5%     50%   97.5% overlap0 f  Rhat n.eff
mean.sj     0.235  0.017   0.204   0.235   0.269    FALSE 1 1.000  7765
mean.sa     0.524  0.020   0.486   0.524   0.563    FALSE 1 1.001  3358
mean.f      3.061  0.496   2.079   3.056   4.042    FALSE 1 1.002  2000
mean.p      0.577  0.028   0.521   0.577   0.632    FALSE 1 1.000  8171
N[1,1]     87.313 52.309   5.000  86.000 186.000    FALSE 1 1.000 30000
N[2,1]     88.064 52.855   5.000  86.000 189.000    FALSE 1 1.000  8663
N[1,2]     60.035 10.002  41.000  60.000  81.000    FALSE 1 1.000 30000
N[2,2]     90.574 15.270  62.000  90.000 123.000    FALSE 1 1.002  3465
[...output truncated...]
N[1,9]     28.276 10.007   9.000  28.000  49.000    FALSE 1 1.001  2601
N[2,9]     40.229  9.823  20.000  40.000  59.000    FALSE 1 1.000 30000
N[1,10]    24.716 10.022   7.000  24.000  46.000    FALSE 1 1.002  3081
N[2,10]    35.632 10.202  15.000  36.000  56.000    FALSE 1 1.000 10527
sigma      34.876 11.609  18.959  32.552  64.186    FALSE 1 1.001  2742
Ntot[1]   175.377 27.164 124.000 175.000 233.000    FALSE 1 1.002  4377
Ntot[2]   150.608 19.421 113.000 150.000 191.000    FALSE 1 1.001  4799
[...output truncated...]
Ntot[9]    68.505 18.161  31.000  69.000 103.000    FALSE 1 1.001  5638
Ntot[10]   60.348 18.787  24.000  60.000  98.000    FALSE 1 1.001  4570
deviance 353.376  4.813 345.128 352.947 364.098    FALSE 1 1.001  3315
```

Both models converge easily. To visualize the results, and in particular the benefits of the structured population counts, we first compare the posterior distributions of the demographic rates.

```
par(mfrow=c(1, 3), cex=1.025, lwd=2)
plot(density(out1$sims.list$mean.sj), type='l', col='blue', main=NA,
xlab='Juvenile survival', axes=FALSE, lwd=2, xlim=c(0.15, 0.45))
lines(density(out2$sims.list$mean.sj), type='l', col='red', lwd=2)
axis(1); axis(2, las=1)
legend('topright', legend=c("Structured counts", "Unstructured counts"),
col=c("blue", "red"), lwd=c(2, 2), bty="n")

plot(density(out1$sims.list$mean.sa), type='l', col='blue', main=NA,
xlab='Adult survival', axes=FALSE, lwd=2)
lines(density(out2$sims.list$mean.sa), type='l', col='red', lwd=2)
axis(1); axis(2, las=1)

plot(density(out1$sims.list$mean.f), type='l', col='blue', main=NA,
xlab='Productivity', axes=FALSE, lwd=2)
lines(density(out2$sims.list$mean.f), type='l', col='red', lwd=2)
axis(1); axis(2, las=1)
```

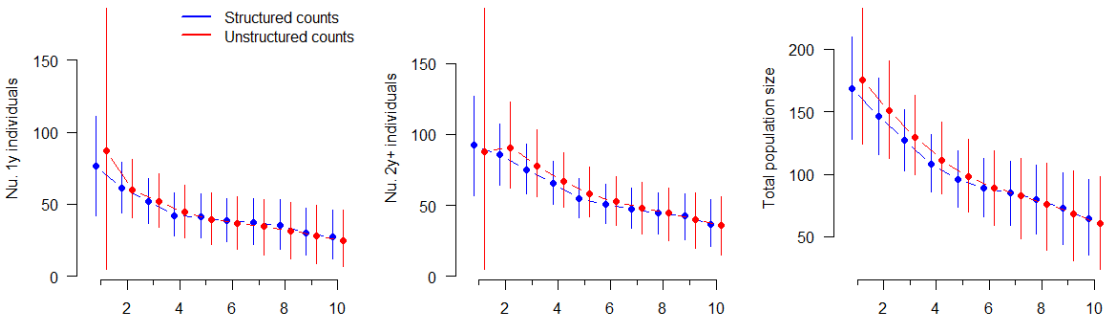


Juvenile survival is nearly identical under the two models, but both adult survival and productivity become more precise when structured counts are used. The CV of adult survival decreased by 8% when structured counts are used compared when we analyzed unstructured counts, for productivity the CV decreased by 30%, thus quite substantially.

Next, we also compare the estimated population sizes under both models.

```
par(mfrow=c(1, 3), cex=1.025, lwd=1.5)
T <- length(out1$mean$Ntot)
d <- 0.2
plot(x=(1:T)-d, y=out1$mean$N[1,], ylim=range(c(out2$q2.5$N[1,],
out2$q97.5$N[1,])), xlim=c(0.5, 10.5), pch=16, type="b", col="blue", ylab="Nu.
1y individuals", xlab=NA, axes=FALSE)
axis(1, at=1:T, tcl=-0.25, labels=NA)
axis(1, at=c(2, 4, 6, 8, 10), labels=c(2, 4, 6, 8, 10))
axis(2, las=1)
segments((1:T)-d, out1$q2.5$N[1,], (1:T)-d, out1$q97.5$N[1,], col="blue")
points(x=(1:T)+d, y=out2$mean$N[1,], pch=16, type="b", col="red")
segments((1:T)+d, out2$q2.5$N[1,], (1:T)+d, out2$q97.5$N[1,], col="red")
legend('topright', legend=c("Structured counts", "Unstructured counts"),
col=c("blue", "red"), lwd=c(2, 2), bty="n")
```

```
plot(x=(1:T)-d, y=out1$mean$N[2,], ylim=range(c(out2$q2.5$N[2,],
out2$q97.5$N[2,])), xlim=c(0.5, 10.5), pch=16, type="b", col="blue", ylab="Nu.
2y+ individuals", xlab=NA, axes=FALSE)
axis(1, at=1:T, tcl=-0.25, labels=NA)
axis(1, at=c(2, 4, 6, 8, 10), labels=c(2, 4, 6, 8, 10))
axis(2, las=1)
segments((1:T)-d, out1$q2.5$N[2,], (1:T)-d, out1$q97.5$N[2,], col="blue")
points(x=(1:T)+d, y=out2$mean$N[2,], pch=16, type="b", col="red")
segments((1:T)+d, out2$q2.5$N[2,], (1:T)+d, out2$q97.5$N[2,], col="red")

plot(x=(1:T)-d, y=out1$mean$Ntot, ylim=range(c(out2$q2.5$Ntot,
out2$q97.5$Ntot)), xlim=c(0.5, 10.5), pch=16, type="b", col="blue", ylab="Total
population size", xlab=NA, axes=FALSE)
axis(1, at=1:T, tcl=-0.25, labels=NA)
axis(1, at=c(2, 4, 6, 8, 10), labels=c(2, 4, 6, 8, 10))
axis(2, las=1)
segments((1:T)-d, out1$q2.5$Ntot, (1:T)-d, out1$q97.5$Ntot, col="blue")
points(x=(1:T)+d, y=out2$mean$Ntot, pch=16, type="b", col="red")
segments((1:T)+d, out2$q2.5$Ntot, (1:T)+d, out2$q97.5$Ntot, col="red")
```



The uncertainties in the stage structured population sizes as well as in the total population size are reduced when the structured counts are used compared to when the IPM is applied to the unstructured counts. The difference is largest in the first year for the stage-specific population sizes.

This exercise illustrates the benefits of using structured counts: the precision of demographic rates and of structured population sizes increases. We expect that the gain in precision due to the use of structured counts increases with increasing complexity of the life cycle and also likely with the dimension of the structured counts. Also, we intuit that the smaller the sample size, the more will structured counts proportionally increase the precision of the estimates compared to the analysis of unstructured counts (and here is another paper that could be written using a simulation study ...).

Exercise 2

Investigate the effect of the prior for the initial population sizes on parameter estimates, i.e., do a prior sensitivity analysis. Use the data set `woodchat6` for this purpose and study i) how sensitive parameter estimates are to alternative priors, and ii) whether prior sensitivity changes

if all data sets (capture-recapture, population counts, productivity) or only capture-recapture data and population counts are used.

Solution: We must decide which population model to use and here favor again the prebreeding 'census', female-based model. We also assume that productivity does not depend on female age. To keep the model simple, we further assume that all demographic rates are constant over time, but we will include demographic stochasticity. Then we need to think about the choice of priors for the initial population sizes. We first note that the observed count in year 1 is 110 breeding pairs. As the stable stage distribution is more or less 1:1 given the demographic rates and the prebreeding life-cycle (to see that you must calculate the stable stage distribution; see, e.g., Exercise 3.1), we expect there to be about 55 1-year old and 55 older females in the population. Thinking about the size of each stage class is important because we must place priors on these parameters. We try out four different priors. First, we use a discrete uniform with a wide range {1, 300} with the intention to be vague. Second, we use again a discrete uniform, but this time choose it to be slightly informative and hence use a narrower range, i.e. {40, 70} for each stage class (note that both numbers are centered on the expected value of 55). Third, we use again a discrete uniform, but this time choose it to be asymmetric around 55, i.e. {50, 80}. Finally, we use a Poisson prior with expected value of 55.

We load the data and fit the four different models.

```
# Load woodchat shrike data
library(IPMbook)
data(woodchat6)

# Transform capture-recapture data into the m-array format
shrike.marray <- marrayAge(woodchat6$ch, woodchat6$age)
```

A: Discrete uniform prior: ~dUnif(1, 300)

```
# Bundle data
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=sum(woodchat6$J), B=sum(woodchat6$B),
C=woodchat6$count, pNinit=dUnif(1, 300))


# Specify the model in BUGS language
cat(file="model3.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  p[t] <- mean.p
}

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma ~ dunif(0.5, 100)
tau <- pow(sigma, -2)
```

```
# State-space model for population count data
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

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

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

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

# Productivity data (Poisson regression model)
J ~ dpois(mean.f * B)    # as constant over time, no loop necessary

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

We prepare the other necessary ingredients for the analysis in JAGS and run the model.

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

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

# MCMC settings
ni <- 20000; nb <- 10000; nc <- 3; nt <- 3; na <- 3000

# Call JAGS, check convergence and summarize posteriors
out3 <- jags(jags.data, inits, parameters, "model3.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out3)     # Not shown
print(out3, 3)
```

B: Discrete uniform prior: ~dUnif(40, 70)

```
# Bundle data
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=sum(woodchat6$J), B=sum(woodchat6$B),
C=woodchat6$count, pNinit=dUnif(40, 70))

# Call JAGS, check convergence and summarize posteriors
out4 <- jags(jags.data, inits, parameters, "model3.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out4)     # Not shown
print(out4, 3)
```

C: Discrete uniform prior: ~dUnif(50, 80)

```
# Bundle data
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=sum(woodchat6$J), B=sum(woodchat6$B),
C=woodchat6$count, pNinit=dUnif(50, 80))

# Call JAGS, check convergence and summarize posteriors
out5 <- jags(jags.data, inits, parameters, "model3.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out5)     # Not shown
print(out5, 3)
```

D: Poisson prior: ~dPois(55)

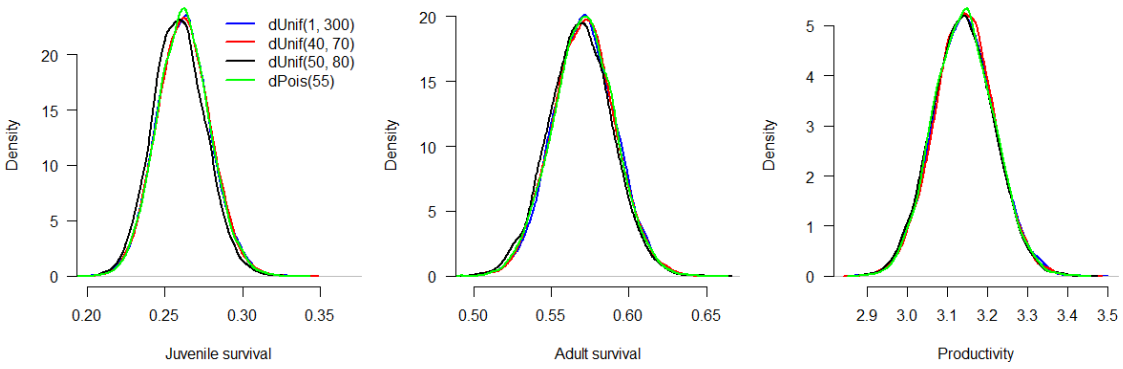
```
# Bundle data
jags.data <- list(marr.j=shrike.marray[,,1], marr.a=shrike.marray[,,2],
n.occasions=dim(shrike.marray)[2], rel.j=rowSums(shrike.marray[,,1]),
rel.a=rowSums(shrike.marray[,,2]), J=sum(woodchat6$J), B=sum(woodchat6$B),
C=woodchat6$count)


# Specify the model in BUGS language
cat(file="model4.txt", "
model {
# Priors and linear models
for (t in 1:(n.occasions-1)){
  sj[t] <- mean.sj
```

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

mean.sj ~ dunif(0, 1)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.f ~ dunif(0, 10)

sigma ~ dunif(0.5, 100)
tau <- pow(sigma, -2)

# State-space model for population count data
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dpois(55)
N[2,1] ~ dpois(55)

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

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

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

# Productivity data (Poisson regression model)
J ~ dpois(mean.f * B)    # as constant over time, no loop necessary

# Derived parameters
# Total population size
```

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

# Call JAGS, check convergence and summarize posteriors
out6 <- jags(jags.data, inits, parameters, "model4.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out5)    # Not shown
print(out5, 3)
```

To assess the impact of the priors on the parameter estimates, we first compare the posterior distributions of the demographic rates under the four models.

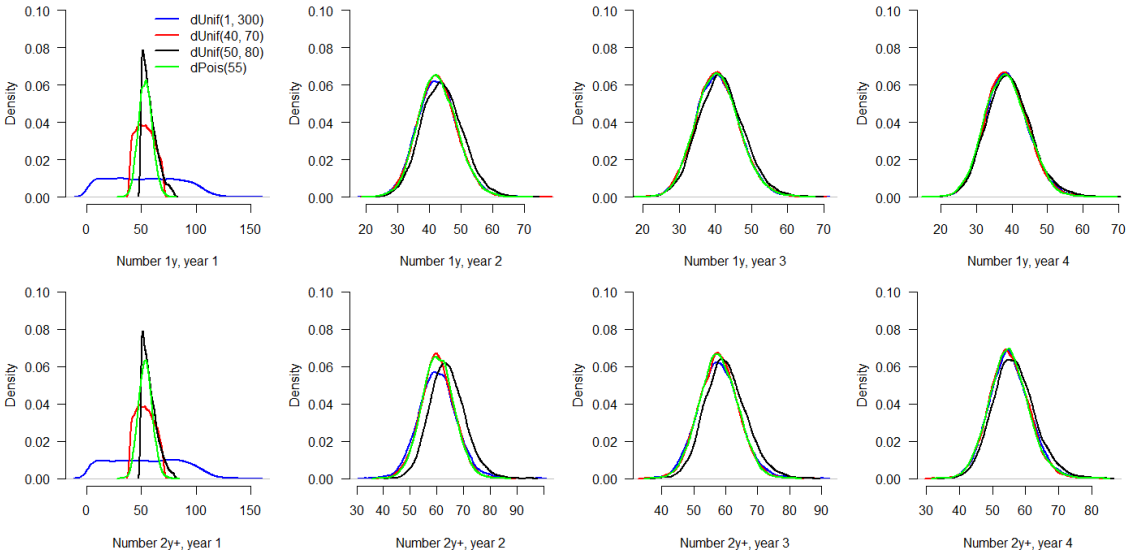
```
par(mfrow=c(1, 3), cex=1.025, lwd=1.5)
plot(density(out3$sims.list$mean.sj), type='l', col='blue', main=NA,
xlab='Juvenile survival', axes=FALSE, lwd=2, xlim=c(0.2, 0.37))
lines(density(out4$sims.list$mean.sj), type='l', col='red', lwd=2)
lines(density(out5$sims.list$mean.sj), type='l', col='black', lwd=2)
lines(density(out6$sims.list$mean.sj), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)
legend('topright', legend=c("dUnif(1, 300)", "dUnif(40, 70)", "dUnif(50, 80)",
"dPois(55)"), col=c("blue", "red", "black", "green"), lwd=c(2, 2), bty="n")

plot(density(out3$sims.list$mean.sa), type='l', col='blue', main=NA,
xlab='Adult survival', axes=FALSE, lwd=2)
lines(density(out4$sims.list$mean.sa), type='l', col='red', lwd=2)
lines(density(out5$sims.list$mean.sa), type='l', col='black', lwd=2)
lines(density(out6$sims.list$mean.sa), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$mean.f), type='l', col='blue', main=NA,
xlab='Productivity', axes=FALSE, lwd=2)
lines(density(out4$sims.list$mean.f), type='l', col='red', lwd=2)
lines(density(out5$sims.list$mean.f), type='l', col='black', lwd=2)
lines(density(out6$sims.list$mean.f), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)
```



We see that in this case the four different priors for the initial population size did not have any relevant impact on the posteriors of the demographic rates.

Next, we also compare the posteriors of the stage-structed population size in the first four years.

```
par(mfrow=c(2, 4), cex=1.025, lwd=1.5, mar=c(4, 4, 1, 1))
plot(density(out3$sims.list$N[,1,1]), type='l', col='blue', main=NA,
xlab='Number 1y, year 1', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,1,1]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,1,1]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,1,1]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)
legend('topright', legend=c("dUnif(1, 300)", "dUnif(40, 70)", "dUnif(50, 80)",
"dPois(55)"), col=c("blue", "red", "black", "green"), lwd=c(2, 2), bty="n")

plot(density(out3$sims.list$N[,1,2]), type='l', col='blue', main=NA,
xlab='Number 1y, year 2', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,1,2]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,1,2]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,1,2]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,1,3]), type='l', col='blue', main=NA,
xlab='Number 1y, year 3', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,1,3]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,1,3]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,1,3]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,1,4]), type='l', col='blue', main=NA,
xlab='Number 1y, year 4', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,1,4]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,1,4]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,1,4]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,2,1]), type='l', col='blue', main=NA,
xlab='Number 2y+, year 1', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,2,1]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,2,1]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,2,1]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,2,2]), type='l', col='blue', main=NA,
xlab='Number 2y+, year 2', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,2,2]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,2,2]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,2,2]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,2,3]), type='l', col='blue', main=NA,
xlab='Number 2y+, year 3', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,2,3]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,2,3]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,2,3]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)

plot(density(out3$sims.list$N[,2,4]), type='l', col='blue', main=NA,
xlab='Number 2y+, year 4', axes=FALSE, lwd=2, ylim=c(0, 0.1))
lines(density(out4$sims.list$N[,2,4]), type='l', col='red', lwd=2)
lines(density(out5$sims.list$N[,2,4]), type='l', col='black', lwd=2)
lines(density(out6$sims.list$N[,2,4]), type='l', col='green', lwd=2)
axis(1); axis(2, las=1)
```



Here, now, we clearly see an effect of the priors on the posteriors of the stage-structured population sizes in the first year, but there was hardly any effect of the four choices of prior in later years. Hence, we stress again that generally the estimates of the (stage-structured) population sizes in the initial year are typically much more uncertain and less reliable than estimates for later years. This has to do with the Markovian structure of the population model: all years except for the first can exploit some of the information that stems from the preceding years via that autoregressive structure.

Prior sensitivity analyses are warranted whenever you are uncertain about the impact of the prior on an estimate and especially if you think that a parameter may be 'weak', i.e., not be well determined by the available information in your data.

Chapter 7

Exercise 1

Established frequentist GoF tests for CJS model (Burnham et al., 1987) are based on summaries of the capture histories. There are several types of such tests which basically assess whether the survival and recapture probabilities are the same in all marked individuals (e.g., tests 3.SR, 2.CT). These tests are specific to certain types of model violations, e.g., the occurrence of transients or trap-dependence and are therefore particularly useful. For each of these tests the data are tabulated in specific ways and it is tested by χ^2-tests whether the tabulated numbers deviate from expected numbers that are calculated under the hypothesis of homogeneous rates. We can use these tests either directly, using the R package `R2ucare` (Gimenez et al., 2018), or implement them into a Bayesian assessment. Assess the goodness-of-fit of the CJS model fitted to the capture-recapture data from the `woodchat5` data sets using these specific tests in the Bayesian framework, i.e., do posterior predictive checks using these tests as discrepancy measures.

Solution. We need to simulate data under the fitted model and then apply the specific tests to each of the simulated data sets. This then results in a reference distribution of the test statistic under the fitting model, which can be compared with the test statistic obtained from fitting the model to the actual data set. Simulation of data sets is straightforward in BUGS, but the computation of the test statistics is somewhat complicated to program in BUGS, although not totally impossible. Thus, here we simulate replicate data sets in JAGS and save them and then compute the test statistics for each of them outside of JAGS after the analysis by applying the specific tests available in package `R2ucare`. Since program U-care conducts the tests for the data that must be in the format of capture-histories (and not in the m-array format), we use the state-space formulation of the CJS model – this then allows the simulation of capture history data sets in a straightforward manner.

The capture-recapture data for the woodchat shrike include individuals marked as juveniles (nestlings) and adults, and thus we expect that survival has an age structure. When an existing age structure is not accounted for in a CJS model, the test 3.SR in particular is sensitive, and we expect a 'significant' result. For our demonstration, we first fit to the complete data set such a basic model with time-dependent survival and recapture probabilities.

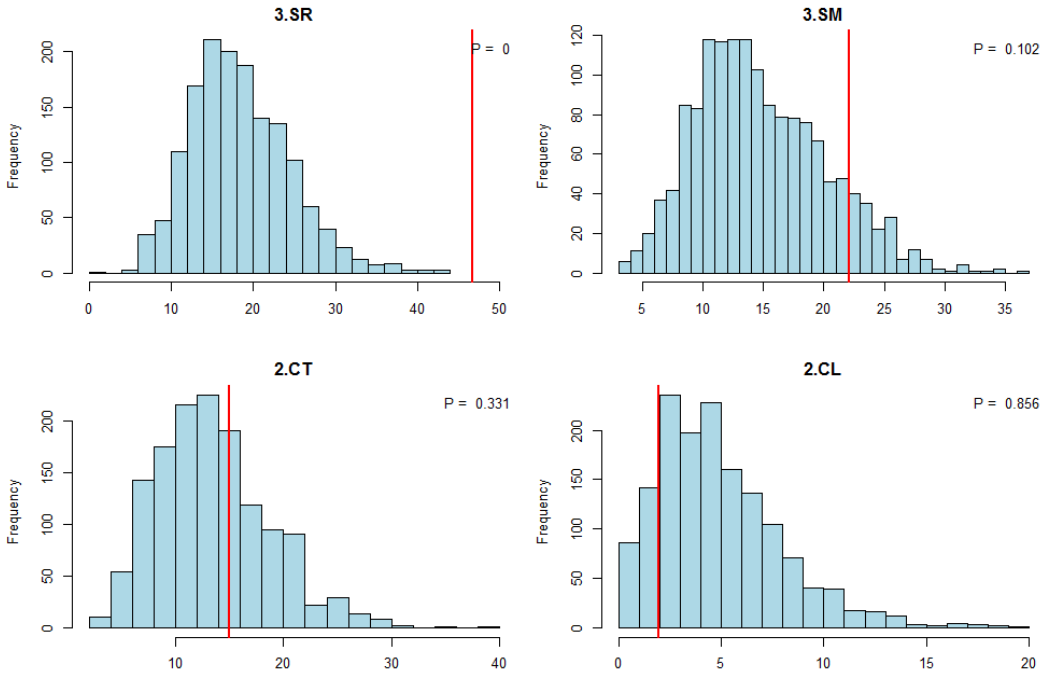
```
# Load the data set
library(IPMbook)
data(woodchat5)
ch <- woodchat5$ch

# Create vector with occasion of marking
f <- apply(ch, 1, getFirst)

# Remove individuals marked at last occasion (reduce size of data set)
ch <- ch[f<20,]
f <- f[f<20]

# Bundle data
jags.data <- list(y=ch, f=f, n.occasions=ncol(ch), n.ind=nrow(ch))

# Specify the CJS model in BUGS language
```

```
cat(file="model1.txt", "
model {
# Priors
for (t in 1:(n.occasions-1)){
  phi[t] ~ dunif(0, 1)
  p[t] ~ dunif(0, 1)
}

# Likelihood
for (i in 1:n.ind){
  # Define latent state at first capture
  z[i,f[i]] <- 1
  z.rep[i,f[i]] <- 1        # replicate z (true state)
  y.rep[i,f[i]] <- 1        # replicate y (data)
  for (t in (f[i]+1):n.occasions){
    # State process
    z[i,t] ~ dbern(phi[t-1] * z[i,t-1])
    z.rep[i,t] ~ dbern(phi[t-1] * z.rep[i,t-1]) # replicate y (data)
    # Observation process
    y[i,t] ~ dbern(p[t-1] * z[i,t])
    y.rep[i,t] ~ dbern(p[t-1] * z.rep[i,t])     # replicate z (true state)
  } #t
} #i
}
")


# Initial values
inits <- function(){list(z=zInit(ch))}

# Parameters monitored
parameters <- c("phi", "p", "y.rep")

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

# Call JAGS (ART 5 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)    # Not shown
print(out1)        # Not shown
```

With the chosen MCMC settings we obtain 1500 replicated data sets, i.e., capture-histories of the same number of animals as exist in our actual data set. To each of them we apply the GoF tests provided in package `R2ucare`, save the test-statistics (Chi2 values), and produce plots.

```
# Load library
library(R2ucare)

# Replace the NAs in the simulated data sets by 0
out1$sims.list$y.rep[is.na(out1$sims.list$y.rep)] <- 0

# Perform U-CARE GoF for each replicated data set
chi.3sr <- chi.3sm <- chi.2cl <- chi.2ct <- numeric()
for (i in 1:dim(out1$sims.list$y.rep)[1]){
  chi.3sr[i] <- test3sr(out1$sims.list$y.rep[i,,], rep(1, nrow(ch)))$test3sr[1]
  chi.3sm[i] <- test3sm(out1$sims.list$y.rep[i,,], rep(1, nrow(ch)))$test3sm[1]
  chi.2ct[i] <- test2ct(out1$sims.list$y.rep[i,,], rep(1, nrow(ch)))$test2ct[1]
  chi.2cl[i] <- test2cl(out1$sims.list$y.rep[i,,], rep(1, nrow(ch)))$test2cl[1]
}
```

```
# Perform U-CARE GoF for each the real (actual) data set
chi.3sr.actual <- test3sr(ch, rep(1, nrow(ch)))$test3sr[1]
chi.3sm.actual <- test3sm(ch, rep(1, nrow(ch)))$test3sm[1]
chi.2ct.actual <- test2ct(ch, rep(1, nrow(ch)))$test2ct[1]
chi.2cl.actual <- test2cl(ch, rep(1, nrow(ch)))$test2cl[1]

# Produce plots
par(mfrow=c(2, 2), mar=c(4, 4, 2, 1))
hist(chi.3sr, nclass=25, col="lightblue", main="3.SR", xlab=NA, xlim=c(0, 50))
abline(v=chi.3sr.actual, col="red", lwd=2)
p.3sr <- round(mean((chi.3sr.actual - chi.3sr) < 0, na.rm=TRUE), 3)
legend("topright", legend=paste("P = ", p.3sr), bty="n")

hist(chi.3sm, nclass=25, col="lightblue", main="3.SM", xlab=NA)
abline(v=chi.3sm.actual, col="red", lwd=2)
p.3sm <- round(mean((chi.3sm.actual - chi.3sm) < 0, na.rm=TRUE), 3)
legend("topright", legend=paste("P = ", p.3sm), bty="n")

hist(chi.2ct, nclass=25, col="lightblue", main="2.CT", xlab=NA)
abline(v=chi.2ct.actual, col="red", lwd=2)
p.2ct <- round(mean((chi.2ct.actual - chi.2ct) < 0, na.rm=TRUE), 3)
legend("topright", legend=paste("P = ", p.2ct), bty="n")

hist(chi.2cl, nclass=25, col="lightblue", main="2.CL", xlab=NA)
abline(v=chi.2cl.actual, col="red", lwd=2)
p.2cl <- round(mean((chi.2cl.actual - chi.2cl) < 0, na.rm=TRUE), 3)
legend("topright", legend=paste("P = ", p.2cl), bty="n")
```



The plots show the distribution of the four test statistics under a fitting model (light-blue) and the observed test statistics computed from our data (red vertical lines). For a fitting model we would expect the red line to lie somewhere "in the middle" of the light-blue distribution, i.e., close to its peak. The P values indicated in each panel represent the probability that the observed test statistics is smaller than the simulated test statistics or, equivalently, the probability to

obtain a more extreme (i.e., greater) value of the test statistic by chance (which means, under the null hypothesis of a fitting model).

Here, we see that the observed 3.SR test statistic is clearly different from the expected value of the test statistic under a fitting model, while for the remaining three tests there is not such a large discrepancy between the expectation under a fitting model and the observed value of the statistic. Test 3.SR is sensitive to missing age-effects on survival – and hence it doesn't come as a surprise that this component indicates lack of fit of our model. The reason is that there really is an age structure in survival, but our model does not accommodate it.

We now could change the model and then see whether the GoF will be better for an alternative model.

Chapter 8

Exercise 1

Using the red-backed shrike data set, fit an IPM that induces a threshold model (2 levels) for density dependence in productivity. Try to specify the model in such a way that the threshold is estimated.

Solution: To implement a threshold for density dependence we can use the `step` function in JAGS. To avoid possible label switching (see Chapter 14 for more on that topic), we estimate productivity for population size above the threshold as one parameter, and the *difference* between high and low productivity as another. The latter can then be constrained to be positive by the prior specification; this prevents label switching in the simple case when there are only two groups. For ease of demonstration, we will not add any temporal random variation in productivity; hence, in our model variation in productivity will be due only to this specific form of density dependence. We load the data, write the model, define the usual specification needed to run the model, and then launch JAGS to experience its wonderful power.

```
# Load the red-backed shrike data
library(IPMbook)
data(redbacked)

# Bundle data
jags.data <- with(redbacked, list(n.occasions=ncol(marr.a), marr.j=marr.j,
marr.a=marr.a, rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), C=count, J=J, B=B,
pNinit=dUnif(1, 50)))


# Write JAGS model file
cat(file="model1.txt", "
model {
# Priors and linear models
# Models for demographic rates
for (t in 1:(n.occasions-1)){
  logit(phij[t]) <- logit.b0.phij + err.phij[t]
  err.phij[t] ~ dnorm(0, tau.phij)
  logit(phia[t]) <- logit.b0.phia + err.phia[t]
  err.phia[t] ~ dnorm(0, tau.phia)
  logit(pj[t]) <- logit.b0.pj + err.pj[t]
  err.pj[t] ~ dnorm(0, tau.pj)
  logit(pa[t]) <- logit.b0.pa + err.pa[t]
  err.pa[t] ~ dnorm(0, tau.pa)
}
for (t in 1:n.occasions){
  f[t] <- fc[1] + d * step(K - N[t])
  log(omega[t]) <- log.b0.om + err.om[t]
  err.om[t] ~ dnorm(0, tau.om)
}

# Priors for variance parameters (hyperparameters)
tau.phij <- pow(sigma.phij, -2)
sigma.phij ~ dunif(0, 10)
sigma2.phij <- pow(sigma.phij, 2)
tau.phia <- pow(sigma.phia, -2)
sigma.phia ~ dunif(0, 10)
sigma2.phia <- pow(sigma.phia, 2)
```

```
tau.om <- pow(sigma.om, -2)
sigma.om ~ dunif(0, 10)
sigma2.om <- pow(sigma.om, 2)
tau.pj <- pow(sigma.pj, -2)
sigma.pj ~ dunif(0, 10)
sigma2.pj <- pow(sigma.pj, 2)
tau.pa <- pow(sigma.pa, -2)
sigma.pa ~ dunif(0, 10)
sigma2.pa <- pow(sigma.pa, 2)
tau ~ dgamma(0.001, 0.001)
sigma2 <- 1 / tau

# Priors for the mean of some demographic rates and resighting rates
b0.phij ~ dunif(0, 1)
b0.phia ~ dunif(0, 1)
b0.om ~ dunif(0, 75)
b0.pj ~ dunif(0, 1)
b0.pa ~ dunif(0, 1)

# Back-transformations
logit.b0.phij <- logit(b0.phij)
logit.b0.phia <- logit(b0.phia)
log.b0.om <- log(b0.om)
logit.b0.pj <- logit(b0.pj)
logit.b0.pa <- logit(b0.pa)

# Priors for productivity
fc[1] ~ dunif(0, 10)              # Productivity when N > K
d ~ dnorm(0, 0.1)T(0,)            # Difference in prod. when N < K
fc[2] <- fc[1] + d                # Resulting productivity when N < K

# Prior for threshold
K ~ dunif(1, 100)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: discrete uniform priors
R[1] ~ dcat(pNinit)       # Local recruits
S[1] ~ dcat(pNinit)       # Surviving adults
I[1] ~ dpois(omega[1])    # Immigrants

# Process model over time: our model of population dynamics
for (t in 2:n.occasions){
  R[t] ~ dpois(f[t-1]/2 * phij[t-1] * N[t-1])      # No. local recruits
  S[t] ~ dbin(phia[t-1], N[t-1])                   # No. surviving adults
  I[t] ~ dpois(omega[t])                           # No. immigrants
}

# Observation model
for (t in 1:n.occasions){
  N[t] <- S[t] + R[t] + I[t]            # Total number of breeding females
  logN[t] <- log(N[t])
  C[t] ~ dlnorm(logN[t], tau)
}

# Capture-recapture data (CJS model with multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
  marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
  marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
}
# Define the cell probabilities of the m-arrays
for (t in 1:(n.occasions-1)){
```

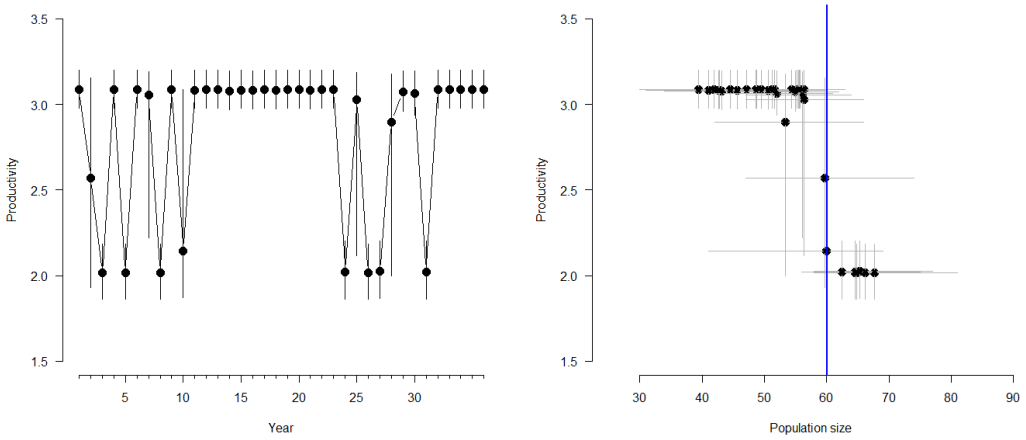
```
  # Main diagonal
  qj[t] <- 1-pj[t]      # Probability of non-recapture (juv)
  qa[t] <- 1-pa[t]      # Probability of non-recapture (ad)
  pr.j[t,t] <- phij[t]*pj[t]
  pr.a[t,t] <- phia[t]*pa[t]
  # Above main diagonal
  for (j in (t+1):(n.occasions-1)){
    pr.j[t,j] <- phij[t]*prod(phia[(t+1):j])*qj[t]*prod(qa[t:(j-
1)])*pa[j]/qa[t]
    pr.a[t,j] <- prod(phia[t:j])*prod(qa[t:(j-1)])*pa[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr.j[t,j] <- 0
    pr.a[t,j] <- 0
  } #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
  pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
} #t

# Productivity data (Poisson regression model)
for (t in 1:n.occasions){
  J[t] ~ dpois(B[t] * f[t])
}

# Derived parameters
# Immigration rate
for (t in 1:(n.occasions-1)){
  imm_rate[t] <- I[t+1] / N[t]
}
}
")


# Define initial values
inits <- function() {list(b0.om=runif(1,0,1), b0.pj=runif(1,0.2,0.7),
b0.pa=runif(1,0.1,0.6), sigma.pa=runif(1,0,1), I=rep(10, 36),
b0.phij=runif(1,0,0.2))}

# Define parameters to be monitored
parameters <- c("phij", "phia", "f", "omega", "imm_rate", "b0.phij", "b0.phia",
"b0.om", "b0.pj", "b0.pa", "sigma2.phij", "sigma2.phia", "sigma2.om",
"sigma2.pj", "sigma2.pa", "R", "S", "I", "N", "pa", "pj", "sigma2", "fc", "K",
"d")

# MCMC settings
ni <- 30000; nb <- 10000; nc <- 3; nt <- 2; na <- 5000 # wonderful convergence
ni <- 3000; nb <- 1000; nc <- 3; nt <- 1; na <- 1000   # quick and dirty

# Call JAGS (ART 50 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)    # Not shown
print(out1)

JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 30000 iterations,
adaptation = 5000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 2,
```

```
yielding 30000 total samples from the joint posterior.
MCMC ran for 65.731 minutes at time 2021-06-11 09:45:21.

                mean     sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
phij[1]        0.064  0.024    0.026    0.061    0.121    FALSE 1 1.002  5821
phij[2]        0.072  0.027    0.033    0.067    0.141    FALSE 1 1.000  7634
[...output truncated...]
phij[34]       0.059  0.019    0.028    0.058    0.103    FALSE 1 1.001 18866
phij[35]       0.061  0.022    0.026    0.059    0.113    FALSE 1 1.000 15040
phia[1]        0.358  0.060    0.241    0.357    0.481    FALSE 1 1.001 27002
phia[2]        0.325  0.057    0.209    0.326    0.434    FALSE 1 1.003   840
[...output truncated...]
phia[34]       0.383  0.053    0.283    0.380    0.496    FALSE 1 1.001  6155
phia[35]       0.381  0.058    0.274    0.377    0.504    FALSE 1 1.001 22222
f[1]           3.085  0.056    2.977    3.084    3.198    FALSE 1 1.001  4387
f[2]           2.570  0.514    1.930    2.962    3.156    FALSE 1 1.000 30000
[...output truncated...]
f[35]          3.085  0.056    2.977    3.084    3.198    FALSE 1 1.001  4387
f[36]          3.085  0.056    2.977    3.084    3.198    FALSE 1 1.001  4387
omega[1]      28.281  4.334   19.212   28.377   37.192    FALSE 1 1.001 30000
omega[2]      31.114  5.165   23.218   30.203   44.331    FALSE 1 1.005   848
[...output truncated...]
omega[35]     29.053  3.968   21.217   28.940   37.762    FALSE 1 1.003  8882
omega[36]     28.893  3.997   21.037   28.809   37.518    FALSE 1 1.002  7753
imm_rate[1]    0.685  0.166    0.415    0.667    1.068    FALSE 1 1.001  7739
imm_rate[2]    0.628  0.140    0.397    0.613    0.945    FALSE 1 1.002  1932
[...output truncated...]
imm_rate[34]   0.523  0.101    0.339    0.518    0.735    FALSE 1 1.001  6390
imm_rate[35]   0.518  0.104    0.328    0.510    0.740    FALSE 1 1.000 12654
b0.phij        0.058  0.009    0.041    0.057    0.078    FALSE 1 1.001  3100
b0.phia        0.364  0.022    0.322    0.363    0.408    FALSE 1 1.000  8476
b0.om         28.914  1.912   25.181   28.904   32.780    FALSE 1 1.000  5880
b0.pj          0.439  0.068    0.312    0.437    0.582    FALSE 1 1.001  3558
b0.pa          0.649  0.039    0.572    0.649    0.725    FALSE 1 1.000  6431
sigma2.phij    0.208  0.168    0.003    0.174    0.610    FALSE 1 1.003  1860
sigma2.phia    0.104  0.073    0.005    0.090    0.283    FALSE 1 1.006   350
sigma2.om      0.024  0.028    0.000    0.014    0.102    FALSE 1 1.017   206
sigma2.pj      0.362  0.594    0.001    0.165    1.703    FALSE 1 1.043   579
sigma2.pa      0.083  0.121    0.000    0.038    0.413    FALSE 1 1.009  1968
R[1]          12.087  8.110    1.000   11.000   30.000    FALSE 1 1.000  4663
R[2]           5.590  3.110    1.000    5.000   13.000    FALSE 1 1.001  5079
[...output truncated...]
R[35]          5.049  2.679    1.000    5.000   11.000    FALSE 1 1.000 30000
R[36]          5.124  2.787    1.000    5.000   11.000    FALSE 1 1.000 18768
S[1]          12.274  8.194    1.000   11.000   31.000    FALSE 1 1.001  4670
S[2]          19.617  4.855   11.000   20.000   29.000    FALSE 1 1.000 13419
[...output truncated...]
S[35]         21.323  4.422   13.000   21.000   30.000    FALSE 1 1.001  3206
S[36]         20.853  4.490   12.000   21.000   30.000    FALSE 1 1.001  2730
I[1]          26.833  6.274   15.000   27.000   40.000    FALSE 1 1.000 18028
I[2]          34.526  6.656   22.000   34.000   49.000    FALSE 1 1.002  2408
[...output truncated...]
I[35]         28.859  4.975   19.000   29.000   39.000    FALSE 1 1.001  6944
I[36]         28.436  5.036   19.000   28.000   38.000    FALSE 1 1.000 30000
N[1]          51.193  5.559   40.000   52.000   61.000    FALSE 1 1.001  4344
N[2]          59.732  6.728   47.000   60.000   74.000    FALSE 1 1.001  2151
[...output truncated...]
N[35]         55.231  4.143   46.000   56.000   62.000    FALSE 1 1.001  2920
N[36]         54.413  4.440   45.000   55.000   62.000    FALSE 1 1.000  4058
pa[1]          0.634  0.071    0.472    0.640    0.761    FALSE 1 1.000 10112
pa[2]          0.635  0.073    0.468    0.641    0.770    FALSE 1 1.000  9740
[...output truncated...]
pa[34]         0.677  0.068    0.559    0.670    0.837    FALSE 1 1.001  4921
pa[35]         0.656  0.067    0.524    0.654    0.801    FALSE 1 1.000 18514
pj[1]          0.467  0.132    0.236    0.452    0.790    FALSE 1 1.001  7380
[...output truncated...]
pj[34]         0.441  0.118    0.217    0.435    0.703    FALSE 1 1.000 30000
pj[35]         0.452  0.127    0.221    0.441    0.757    FALSE 1 1.001 30000
```

```
sigma2          0.029   0.010    0.014    0.027    0.053    FALSE 1 1.001  3826
fc[1]           2.014   0.082    1.862    2.012    2.186    FALSE 1 1.001 10951
fc[2]           3.085   0.056    2.977    3.084    3.198    FALSE 1 1.001  4387
K              60.085   2.576   55.200   59.956   65.435    FALSE 1 1.007   335
d               1.070   0.088    0.895    1.071    1.238    FALSE 1 1.000 30000
deviance     1126.286  15.616 1096.404 1126.154 1157.347    FALSE 1 1.003   845
```

We need to run the model for quite a number of iterations, but for the long MCMC settings above, convergence is pretty good (and usually quite acceptable also for the shorter of the two). We then produce plots to visualize the effect of population size on productivity.

```
# Produce some graphs to visualize the results
par(mfrow=c(1, 2))
T <- length(out1$mean$f)
plot(out1$mean$f, type="b", ylim=c(1.5, 3.5), ylab="Productivity", xlab="Year",
las=1, cex=1.5, axes=FALSE, pch=16)
axis(1, at=1:T, labels=NA, tcl=-0.25)
axis(1, at=c(5, 10, 15, 20, 25, 30), labels=c(5, 10, 15, 20, 25, 30), tcl=-0.5)
axis(2, las=1)
segments(1:T, out1$q2.5$f, 1:T, out1$q97.5$f)

plot(y=out1$mean$f, x=out1$mean$N, type="p", ylim=c(1.5, 3.5), xlim=c(25, 90),
ylab="Productivity", xlab="Population size", las=1, cex=1.5, axes=FALSE,
pch=16)
axis(1); axis(2, las=1)
segments(out1$mean$N, out1$q2.5$f, out1$mean$N, out1$q97.5$f, col="grey70")
segments(out1$q2.5$N, out1$mean$f, out1$q97.5$N, out1$mean$f, col="grey70")
points(y=out1$mean$f, x=out1$mean$N, type="p", pch=16)
abline(v=out1$mean$K, col="blue", lwd=2)
```



The plot of productivity against time shows that in most years productivity was high, i.e., population size was below the threshold at which density dependence was assumed to kick in. When we plot productivity against population size, we recognize that the threshold is at about 60 (you see the exact estimate when you look up on the line for *K* in the posterior summary above). When population size in a year is smaller than that, productivity is high, and *vice versa*.

Given the model we might have expected to get estimates of annual productivity that consist of two values only: one when population size is above and another when it is below the threshold. Looking at the graph (left panel) this is indeed the case for most years, but not for all –

in some we get intermediate productivity values. The reason for this behavior, which may appear counter-intuitive at first, is the following: if population size in a given year is close to the threshold value, there is uncertainty about it, which translates into uncertainty about whether productivity should be high or low. For example if half of the MCMC draws of population size is above and the other half is below the threshold, we will obtain an estimate of productivity (i.e., a posterior mean) that is the mean of high and low productivity (i.e., ~(fc[1] + fc[2]) * 0.5). This behavior can be seen well for years 2 and 28 in the left panel. The corresponding population sizes are right on the blue line (i.e., on the estimated threshold) in the right panel.

Exercise 2

Fit an IPM to the red-backed shrike data set which uses a Gompertz model for density dependence on productivity. How do the estimates differ compared to those of the Ricker model?

Solution: The Ricker model directly uses *N* as a predictor in the model of productivity, while the Gompertz model uses log(*N*). Thus, only a small change in the model code of Section 8.2.2 is necessary. However, we also need to remove density dependence in juvenile and adult survival, so that density dependence is induced only via productivity here. We load the data, write the model, and run it.

```
# Load the red-backed shrike data
library(IPMbook)
data(redbacked)

# Bundle data
jags.data <- with(redbacked, list(n.occasions=ncol(marr.a), marr.j=marr.j,
marr.a=marr.a, rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), C=count, J=J, B=B,
pNinit=dUnif(1, 50)))


# Write JAGS model file
cat(file="model2.txt", "
model {
# Priors and linear models
# Models for demographic rates
for (t in 1:(n.occasions-1)){
  logit(phij[t]) <- logit.b0.phij + err.phij[t]
  err.phij[t] ~ dnorm(0, tau.phij)
  logit(phia[t]) <- logit.b0.phia + err.phia[t]
  err.phia[t] ~ dnorm(0, tau.phia)
  logit(pj[t]) <- logit.b0.pj + err.pj[t]
  err.pj[t] ~ dnorm(0, tau.pj)
  logit(pa[t]) <- logit.b0.pa + err.pa[t]
  err.pa[t] ~ dnorm(0, tau.pa)
}
for (t in 1:n.occasions){
  log(f[t]) <- alpha + beta * log(N[t]) + err.f[t]
  err.f[t] ~ dnorm(0, tau.f)
  log(omega[t]) <- log.b0.om + err.om[t]
  err.om[t] ~ dnorm(0, tau.om)
}

# Priors for variance parameters (hyperparameters)
```

```
tau.phij <- pow(sigma.phij, -2)
sigma.phij ~ dunif(0, 10)
sigma2.phij <- pow(sigma.phij, 2)
tau.phia <- pow(sigma.phia, -2)
sigma.phia ~ dunif(0, 10)
sigma2.phia <- pow(sigma.phia, 2)
tau.f <- pow(sigma.f, -2)
sigma.f ~ dunif(0, 10)
sigma2.f <- pow(sigma.f, 2)
tau.om <- pow(sigma.om, -2)
sigma.om ~ dunif(0, 10)
sigma2.om <- pow(sigma.om, 2)
tau.pj <- pow(sigma.pj, -2)
sigma.pj ~ dunif(0, 10)
sigma2.pj <- pow(sigma.pj, 2)
tau.pa <- pow(sigma.pa, -2)
sigma.pa ~ dunif(0, 10)
sigma2.pa <- pow(sigma.pa, 2)
tau ~ dgamma(0.001, 0.001)
sigma2 <- 1 / tau

# Priors for the mean of demographic and resighting rates
b0.phij ~ dunif(0, 1)
b0.phia ~ dunif(0, 1)
b0.om ~ dunif(0, 75)
b0.pj ~ dunif(0, 1)
b0.pa ~ dunif(0, 1)

# Priors for regression parameters of productivity
alpha ~ dnorm(0, 0.01)    # intercept
beta ~ dunif(-5, 5)       # density dependence

# Back-transformations
logit.b0.phij <- logit(b0.phij)
logit.b0.phia <- logit(b0.phia)
log.b0.om <- log(b0.om)
logit.b0.pj <- logit(b0.pj)
logit.b0.pa <- logit(b0.pa)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: discrete uniform priors
R[1] ~ dcat(pNinit)      # Local recruits
S[1] ~ dcat(pNinit)      # Surviving adults
I[1] ~ dpois(omega[1])   # Immigrants

# Process model over time: our model of population dynamics
for (t in 2:n.occasions){
  R[t] ~ dpois(f[t-1] / 2 * phij[t-1] * N[t-1])    # No. local recruits
  S[t] ~ dbin(phia[t-1], N[t-1])                   # No. surviving adults
  I[t] ~ dpois(omega[t])                           # No. immigrants
}

# Observation model
for (t in 1:n.occasions){
  N[t] <- S[t] + R[t] + I[t]            # Total number of breeding females
  logN[t] <- log(N[t])
  C[t] ~ dlnorm(logN[t], tau)
}

# Capture-recapture data (CJS model with multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
```

```
  marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
  marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
}
# Define the cell probabilities of the m-arrays
for (t in 1:(n.occasions-1)){
  # Main diagonal
  qj[t] <- 1-pj[t]     # Probability of non-recapture (juv)
  qa[t] <- 1-pa[t]     # Probability of non-recapture (ad)
  pr.j[t,t] <- phij[t] * pj[t]
  pr.a[t,t] <- phia[t] * pa[t]
  # Above main diagonal
  for (j in (t+1):(n.occasions-1)){
    pr.j[t,j] <- phij[t] * prod(phia[(t+1):j]) * qj[t]*prod(qa[t:(j-1)]) *
pa[j] / qa[t]
    pr.a[t,j] <- prod(phia[t:j]) * prod(qa[t:(j-1)]) * pa[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr.j[t,j] <- 0
    pr.a[t,j] <- 0
  } #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
  pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
}

# Productivity data (Poisson regression model)
for (t in 1:n.occasions){
  J[t] ~ dpois(B[t] * f[t])
}
}
")

# Define initial values
inits <- function() {list(b0.om=runif(1,0,1), b0.pj=runif(1,0.2,0.7),
b0.pa=runif(1,0.1,0.6), sigma.pa=runif(1,0,1), I=rep(10, 36))}

# Define parameters to be monitored
parameters <- c("phij", "phia", "f", "omega", "b0.phij", "b0.phia", "b0.om",
"b0.pj", "b0.pa", "sigma2.phij", "sigma2.phia", "sigma2.f", "sigma2.om",
"sigma2.pj", "sigma2.pa", "R", "S", "I", "N", "pa", "pj", "sigma2", "alpha",
"beta")

# MCMC settings
ni <- 110000; nb <- 10000; nc <- 3; nt <- 40; na <- 5000 # Good convergence
ni <- 6000; nb <- 1000; nc <- 4; nt <- 5; na <- 1000     # .... dirty

# Call JAGS (ART 150 min), check convergence and summarize posteriors
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out2)    # Not shown
print(out2)
```

The model needs more than 3 hours to run, but then convergence is satisfactory, at least for the longer set of MCMC settings.

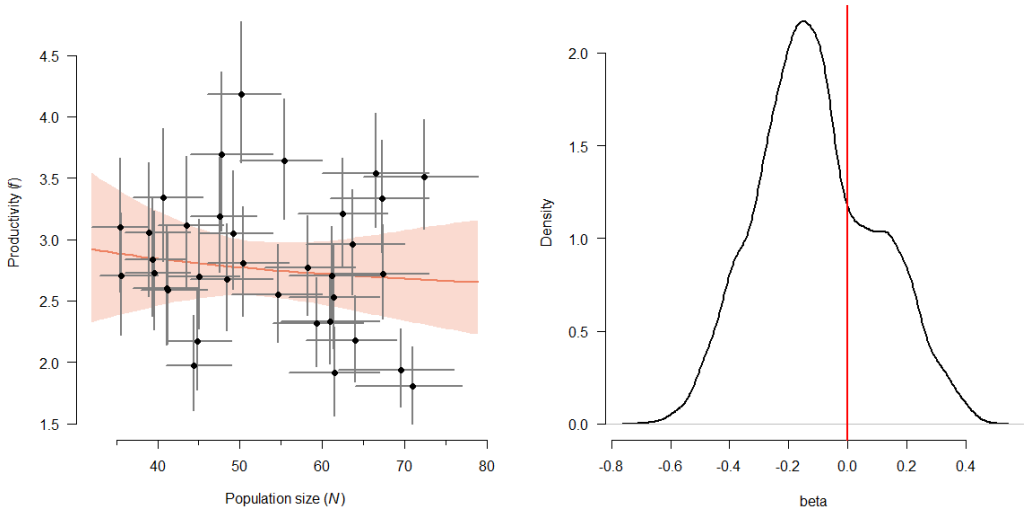
```
               mean      sd     2.5%      25%      50%     75%   97.5% Rhat n.eff overlap0    f
phij[1]      0.0677  0.0266  2.7e-02  4.9e-02   0.0635 8.1e-02 1.3e-01    1  7500        0 1.00
phij[2]      0.0730  0.0276  3.1e-02  5.4e-02   0.0683 8.7e-02 1.4e-01    1  3664        0 1.00
[...output truncated...]
phij[34]     0.0631  0.0215  2.9e-02  4.8e-02   0.0606 7.5e-02 1.1e-01    1  2845        0 1.00
```

```
phij[35]      0.0623  0.0228  2.7e-02  4.6e-02   0.0593 7.4e-02 1.2e-01    1  7500        0 1.00
phia[1]       0.3668  0.0647  2.5e-01  3.2e-01   0.3640 4.1e-01 5.0e-01    1  3277        0 1.00
phia[2]       0.3132  0.0590  2.0e-01  2.7e-01   0.3133 3.5e-01 4.3e-01    1  2546        0 1.00
[...output truncated...]
phia[34]      0.4037  0.0576  3.0e-01  3.6e-01   0.4009 4.4e-01 5.3e-01    1  7500        0 1.00
phia[35]      0.3888  0.0616  2.8e-01  3.5e-01   0.3861 4.3e-01 5.2e-01    1  7500        0 1.00
f[1]          3.6945  0.3303  3.1e+00  3.5e+00   3.6845 3.9e+00 4.4e+00    1  7500        0 1.00
f[2]          2.5260  0.2179  2.1e+00  2.4e+00   2.5214 2.7e+00 3.0e+00    1  7500        0 1.00
[...output truncated...]
f[35]         3.3330  0.2354  2.9e+00  3.2e+00   3.3303 3.5e+00 3.8e+00    1  7500        0 1.00
f[36]         3.0675  0.2266  2.7e+00  2.9e+00   3.0604 3.2e+00 3.5e+00    1  4243        0 1.00
omega[1]     28.1570  3.5040  2.1e+01  2.6e+01  28.2542 3.0e+01 3.5e+01    1  2645        0 1.00
omega[2]     30.4556  3.9252  2.5e+01  2.8e+01  29.7727 3.2e+01 4.0e+01    1  7145        0 1.00
[...output truncated...]
omega[35]    29.4568  3.6065  2.3e+01  2.7e+01  29.1448 3.1e+01 3.8e+01    1  7500        0 1.00
omega[36]    28.7655  3.3942  2.2e+01  2.7e+01  28.6303 3.1e+01 3.6e+01    1  5368        0 1.00
b0.phij       0.0583  0.0097  4.1e-02  5.2e-02   0.0578 6.4e-02 7.9e-02    1  4899        0 1.00
b0.phia       0.3678  0.0227  3.2e-01  3.5e-01   0.3675 3.8e-01 4.1e-01    1  7500        0 1.00
b0.om        28.6192  1.6509  2.5e+01  2.8e+01  28.5958 3.0e+01 3.2e+01    1  7500        0 1.00
b0.pj         0.4352  0.0690  3.1e-01  3.9e-01   0.4321 4.8e-01 5.8e-01    1  7500        0 1.00
b0.pa         0.6426  0.0384  5.7e-01  6.2e-01   0.6431 6.7e-01 7.2e-01    1  7500        0 1.00
sigma2.phij   0.2435  0.1784  7.4e-03  1.2e-01   0.2104 3.3e-01 7.0e-01    1  3812        0 1.00
sigma2.phia   0.1355  0.0762  2.0e-02  8.2e-02   0.1228 1.8e-01 3.2e-01    1  2914        0 1.00
sigma2.f      0.0513  0.0156  2.8e-02  4.0e-02   0.0490 5.9e-02 8.9e-02    1  5939        0 1.00
sigma2.om     0.0142  0.0168  2.6e-05  2.1e-03   0.0084 2.0e-02 5.8e-02    1  1742        0 1.00
sigma2.pj     0.3706  0.5513  4.0e-04  4.4e-02   0.1862 4.8e-01 1.8e+00    1  2077        0 1.00
sigma2.pa     0.0763  0.1069  1.2e-04  8.4e-03   0.0357 1.0e-01 3.8e-01    1  6808        0 1.00
R[1]         10.6944  6.8700  1.0e+00  5.0e+00  10.0000 1.5e+01 2.5e+01    1  7500        0 1.00
R[2]          6.9121  3.5779  1.0e+00  4.0e+00   6.0000 9.0e+00 1.5e+01    1  7500        0 1.00
[...output truncated...]
R[35]         7.8821  3.6549  2.0e+00  5.0e+00   7.0000 1.0e+01 1.6e+01    1  1645        0 1.00
R[36]         6.9624  3.4285  1.0e+00  4.0e+00   7.0000 9.0e+00 1.5e+01    1  7500        0 1.00
S[1]         10.7087  6.7892  1.0e+00  5.0e+00  10.0000 1.5e+01 2.6e+01    1  7500        0 1.00
S[2]         19.2867  4.2764  1.1e+01  1.6e+01  19.0000 2.2e+01 2.8e+01    1  7500        0 1.00
[...output truncated...]
S[35]        28.0371  4.9058  1.9e+01  2.5e+01  28.0000 3.1e+01 3.8e+01    1  7500        0 1.00
S[36]        26.2507  4.9906  1.7e+01  2.3e+01  26.0000 3.0e+01 3.6e+01    1  7500        0 1.00
I[1]         26.4333  5.7133  1.6e+01  2.2e+01  26.0000 3.0e+01 3.8e+01    1  7500        0 1.00
I[2]         35.2045  5.1326  2.5e+01  3.2e+01  35.0000 3.9e+01 4.5e+01    1  7500        0 1.00
[...output truncated...]
I[35]        31.3359  5.1942  2.1e+01  2.8e+01  31.0000 3.5e+01 4.2e+01    1  7500        0 1.00
I[36]        28.8377  5.0180  1.9e+01  2.5e+01  29.0000 3.2e+01 3.9e+01    1  7500        0 1.00
N[1]         47.8364  2.5364  4.4e+01  4.6e+01  48.0000 4.9e+01 5.4e+01    1  3514        0 1.00
N[2]         61.4033  2.7661  5.6e+01  6.0e+01  61.0000 6.3e+01 6.7e+01    1  7500        0 1.00
[...output truncated...]
N[35]        67.2551  2.9951  6.1e+01  6.6e+01  67.0000 6.9e+01 7.3e+01    1  5394        0 1.00
N[36]        62.0508  2.8657  5.6e+01  6.0e+01  62.0000 6.4e+01 6.8e+01    1  7500        0 1.00
pa[1]         0.6280  0.0690  4.7e-01  5.9e-01   0.6329 6.7e-01 7.5e-01    1  7500        0 1.00
pa[2]         0.6331  0.0700  4.8e-01  6.0e-01   0.6363 6.8e-01 7.7e-01    1  2667        0 1.00
[...output truncated...]
pa[34]        0.6675  0.0668  5.5e-01  6.2e-01   0.6610 7.0e-01 8.2e-01    1  7500        0 1.00
pa[35]        0.6471  0.0648  5.2e-01  6.1e-01   0.6457 6.8e-01 7.9e-01    1  6898        0 1.00
pj[1]         0.4593  0.1345  2.2e-01  3.7e-01   0.4452 5.3e-01 7.8e-01    1  7500        0 1.00
pj[2]         0.4315  0.1207  2.0e-01  3.6e-01   0.4265 5.0e-01 7.1e-01    1  6999        0 1.00
[...output truncated...]
pj[34]        0.4357  0.1188  2.1e-01  3.6e-01   0.4276 5.0e-01 7.0e-01    1  7500        0 1.00
pj[35]        0.4506  0.1297  2.2e-01  3.7e-01   0.4392 5.2e-01 7.6e-01    1  6220        0 1.00
sigma2        0.0025  0.0024  3.6e-04  9.7e-04   0.0017 3.2e-03 9.3e-03    1  7500        0 1.00
alpha         1.4330  0.7979 -1.8e-01  8.8e-01   1.5048 2.0e+00 2.9e+00    1    71        1 0.96
beta         -0.1058  0.2009 -4.7e-01 -2.4e-01  -0.1237 3.6e-02 3.0e-01    1    71        1 0.71
deviance    954.0633 33.6468  8.9e+02  9.3e+02 953.6443 9.8e+02 1.0e+03    1  6348        0 1.00
```

The strength of density dependence on productivity (`beta`) is estimated at a negative value, hence, productivity tends to decline as population size increases. Next, we first graph the estimated relationship between population size and productivity and the posterior distribution of `beta`.

```
# Predict the demographic rates
n <- length(out2$mean$N)-1
nx.min <- round(min(out2$q2.5$N[1:n]))
nx.max <- round(max(out2$q97.5$N[1:n]))
nx <- nx.min:nx.max
fec.pred <- matrix(NA, nrow=out2$mcmc.info$n.samples, ncol=length(nx))
for (i in 1:length(nx)){
   fec.pred[,i] <- exp(out2$sims.list$alpha + out2$sims.list$beta * log(nx[i]))
   }
quant <- function (x) quantile(x, c(0.025, 0.975))
```

```
library(scales)
par(mar=c(4, 4.5, 0.5, 2), mfrow=c(1, 2), lwd=2)
plot(NA, ylab=expression('Productivity ('*italic(f)*')'),
xlab=expression('Population size ('*italic(N)*')'), axes=FALSE,
ylim=c(min(out2$q2.5$f), max(out2$q97.5$f)), xlim=c(min(out2$q2.5$N[1:n]),
max(out2$q97.5$N[1:n])))
polygon(x=c(nx, nx.max:nx.min), y=c(apply(fec.pred, 2, quant)[1,],
apply(fec.pred, 2, quant)[2,length(nx):1]), border=NA, col=alpha("salmon2",
0.3))
lines(y=apply(fec.pred, 2, mean), x=nx, lwd=2, col="salmon2")
segments(out2$mean$N[1:n], out2$q2.5$f[1:n], out2$mean$N[1:n],
out2$q97.5$f[1:n], col="grey50")
segments(out2$q2.5$N[1:n], out2$mean$f[1:n], out2$q97.5$N[1:n],
out2$mean$f[1:n], col="grey50")
points(y=out2$mean$f[1:n], x=out2$mean$N[1:n], pch=16)
axis(1)
axis(1, at=c(35, 45, 55, 65, 75), labels=NA, tcl=-0.25)
axis(2, las=1)

plot(density(out2$sims.list$beta), lwd=2, axes=FALSE, main=NA, xlab='beta')
axis(1); axis(2, las=1)
abline(v=0, lwd=2, col='red')
```



In the left panel we see the degree to which productivity declines when population size increases. Compared to the Ricker model that we adopted in the book (see Fig. 8.3C), we don't see much difference. The posterior distribution of the strength of density dependence on productivity (right panel, `beta`) does provide some slight evidence perhaps that density dependence may be negative in this population. The posterior does not look very nicely shaped, the right side of the distribution is not smooth. We note that this parameter was also quite hard to get to chain convergence, and although it is considered to have converged here using our formal criterion of R-hat < 1.1, if this were an analysis for a journal publication, we might run it for 10 times longer (after all, what's 1 month in the lifetime of a scientist?). The dent in the kernel density estimate of the posterior of `beta` is likely the result of one chain that made some excursions far away from the high-density region.

Chapter 9

Exercise 1

Fit an IPM to the hoopoe data set and assess the demographic drivers using the classical LTRE. Develop an IPM in which the temporal correlation among demographic rates is explicitly estimated. Does the inference regarding demographic drivers change compared to the conclusions from Section 9.3?

Solution. We need to change the IPM in such a way that the correlation among demographic rates is estimated as a set of parameters, and then use the estimates obtained in a classical LTRE. To estimate temporal correlation among demographic rates we use the multivariate normal (MVN) distribution, and there are some options regarding the priors that are possible (see Section 6.4). Here we follow the recent recommendations by Riecke et al. (2019) and re-parameterize the variance-covariance matrix in terms of temporal standard deviations and a process correlation coefficient for each pair of rates. JAGS permits direct specification of the variance-covariance matrix of an MVN distribution as `dmnorm.vcov`. We parameterize the variance-covariance matrix in terms of the temporal standard deviations and correlation coefficients and then place suitable uniform priors on both.

We first load the data, write the model, and run it.

```
# Load the hoopoe data
library(IPMbook)
data(hoopoe)

# Produce age-dependent m-arrays
marr <- marrayAge(hoopoe$ch, hoopoe$age, 2)
marr.j <- marr[,,1]
marr.a <- marr[,,2]

# Bundle data
jags.data <- with(hoopoe$reproAgg, list(marr.j=marr.j, marr.a=marr.a,
n.occasions=ncol(marr.j), rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), J1=J1,
B1=B1, J2=J2, B2=B2, count=hoopoe$count, pNinit=dUnif(1, 50)))


# Write JAGS model file
cat(file="model1.txt", "
model {
# Priors and linear models
# For mean parameters
mu[1] <- logit(mean.phij)
mean.phij ~ dunif(0, 1)
mu[2] <- logit(mean.phia)
mean.phia ~ dunif(0, 1)
mu[3] <- log(mean.omega)
mean.omega ~ dunif(0, 3)
mu[4] <- log(mean.f1)
mean.f1 ~ dunif(0, 10)
mu[5] <- log(mean.f2)
mean.f2 ~ dunif(0, 10)
mean.p ~ dunif(0, 1)
mean.pj ~ dunif(0, 1)

# Linear models for demographic parameters
```

```
for (t in 1:(n.occasions-1)){
  logit(phij[t]) <- eps[1,t]
  logit(phia[t]) <- eps[2,t]
  log(omega[t]) <- eps[3,t]
  log(f1[t]) <- eps[4,t]
  log(f2[t]) <- eps[5,t]
  p[t] <- mean.p
  pj[t] <- mean.pj
  eps[1:5,t] ~ dmnorm.vcov(mu[], sigma2[,])
}

# Reparameterize sigma2 in terms of SD and rho
for (i in 1:5){
  sigma2[i,i] <- sigma[i] * sigma[i]
}
for (i in 1:4){
  for (j in (i+1):5){
    sigma2[i,j] <- sigma[i] * sigma[j] * rho[i+j-2]
    sigma2[j,i] <- sigma2[i,j]
  } #j
} #i

# Specify priors for SD and rho
for (i in 1:10){
  sigma[i] ~ dunif(0, 5)
  rho[i] ~ dunif(-1, 1)
}

sigma.res ~ dunif(0.4, 20)      # lower bound >0
tau.res <- pow(sigma.res, -2)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)
N[3,1] ~ dcat(pNinit)

# Process model over time: our model of population dynamics
for (t in 1:(n.occasions-1)){
  N[1,t+1] ~ dpois(f1[t] / 2 * phij[t] * (N[1,t] + N[3,t]) + f2[t] / 2 *
phij[t] * N[2,t])
  N[2,t+1] ~ dbin(phia[t], (N[1,t] + N[2,t] + N[3,t]))
  N[3,t+1] ~ dpois((N[1,t] + N[2,t] + N[3,t]) * omega[t])
}

# Observation model
for (t in 1:n.occasions){
  count[t] ~ dnorm(Ntot[t], tau.res)
  Ntot[t] <- N[1,t] + N[2,t] + N[3,t]    # total population size
}

# Productivity data (Poisson regression model)
for (t in 1:(n.occasions-1)){
  J1[t] ~ dpois(f1[t] * B1[t])
  J2[t] ~ dpois(f2[t] * B2[t])
}

# Capture-recapture data (CJS model with multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
  marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
  marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
```

```
}
# Define the cell probabilities of the m-arrays
for (t in 1:(n.occasions-1)){
  # Main diagonal
  qj[t] <- 1 - pj[t]
  q[t] <- 1 - p[t]
  pr.j[t,t] <- phij[t] * pj[t]
  pr.a[t,t] <- phia[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(n.occasions-1)){
    pr.j[t,j] <- phij[t] * prod(phia[(t+1):j]) * qj[t] * prod(q[t:(j-1)]) *
p[j] / q[t]
    pr.a[t,j] <- prod(phia[t:j]) * prod(q[t:(j-1)]) * p[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr.j[t,j] <- 0
    pr.a[t,j] <- 0
  } #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr.j[t,n.occasions] <- 1 - sum(pr.j[t,1:(n.occasions-1)])
  pr.a[t,n.occasions] <- 1 - sum(pr.a[t,1:(n.occasions-1)])
} #t

# Derived parameters
# Annual population growth rate
for (t in 1:(n.occasions-1)){
  lambda[t] <- Ntot[t+1] / Ntot[t]
}
}
")

# Initial values
inits <- function(){list(mean.phij=runif(1, 0.05, 0.2), mean.phia=runif(1, 0.3,
0.5))}

# Parameters monitored
parameters <- c("mean.phij", "mean.phia", "mean.omega", "mean.f1", "mean.f2",
"sigma2", "rho", "mean.p", "mean.pj", "sigma.res", "phij", "phia", "omega",
"f1", "f2", "N", "Ntot", "lambda")

# MCMC settings
ni <- 30000; nb <- 10000; nc <- 3; nt <- 4; na <- 2000     # Good
ni <- 3000; nb <- 1000; nc <- 3; nt <- 3; na <- 1000       # Quick and dirty

# Call JAGS (ART 20 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)    # Not shown
print(out1)

JAGS output for model 'model1.txt', generated by jagsUI.
Estimates based on 3 chains of 30000 iterations,
adaptation = 2000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 4,
yielding 15000 total samples from the joint posterior.
MCMC ran for 24.842 minutes at time 2021-06-11 11:44:44.

               mean     sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
mean.phij     0.122  0.012   0.099   0.121   0.147    FALSE 1.000 1.005   456
mean.phia     0.365  0.020   0.326   0.364   0.406    FALSE 1.000 1.004   630
```

```
mean.omega     0.289  0.045   0.207   0.288   0.381    FALSE 1.000 1.018   144
mean.f1        5.504  0.302   4.934   5.493   6.136    FALSE 1.000 1.000 15000
mean.f2        6.148  0.400   5.408   6.131   7.000    FALSE 1.000 1.001  2059
sigma2[1,1]    0.090  0.071   0.012   0.073   0.265    FALSE 1.000 1.014   541
sigma2[2,1]    0.000  0.025  -0.055   0.001   0.050     TRUE 0.547 1.004   572
sigma2[3,1]    0.000  0.029  -0.069   0.001   0.052     TRUE 0.441 1.004  1234
sigma2[4,1]    0.014  0.017  -0.017   0.012   0.053     TRUE 0.829 1.002   983
sigma2[5,1]    0.016  0.019  -0.016   0.014   0.061     TRUE 0.844 1.014   217
sigma2[1,2]    0.000  0.025  -0.055   0.001   0.050     TRUE 0.547 1.004   572
sigma2[2,2]    0.042  0.048   0.000   0.027   0.167    FALSE 1.000 1.003  1019
sigma2[3,2]    0.007  0.015  -0.013   0.003   0.041     TRUE 0.829 1.035   198
sigma2[4,2]    0.008  0.012  -0.008   0.006   0.037     TRUE 0.844 1.003  1348
sigma2[5,2]    0.012  0.015  -0.010   0.009   0.050     TRUE 0.852 1.011   218
sigma2[1,3]    0.000  0.029  -0.069   0.001   0.052     TRUE 0.441 1.004  1234
sigma2[2,3]    0.007  0.015  -0.013   0.003   0.041     TRUE 0.829 1.035   198
sigma2[3,3]    0.063  0.112   0.000   0.021   0.387    FALSE 1.000 1.011   836
sigma2[4,3]    0.010  0.016  -0.010   0.006   0.052     TRUE 0.852 1.008   302
sigma2[5,3]    0.009  0.024  -0.034   0.004   0.069     TRUE 0.714 1.013   316
sigma2[1,4]    0.014  0.017  -0.017   0.012   0.053     TRUE 0.829 1.002   983
sigma2[2,4]    0.008  0.012  -0.008   0.006   0.037     TRUE 0.844 1.003  1348
sigma2[3,4]    0.010  0.016  -0.010   0.006   0.052     TRUE 0.852 1.008   302
sigma2[4,4]    0.039  0.020   0.015   0.035   0.090    FALSE 1.000 1.001  7424
sigma2[5,4]    0.026  0.016   0.003   0.023   0.064    FALSE 0.986 1.002 15000
sigma2[1,5]    0.016  0.019  -0.016   0.014   0.061     TRUE 0.844 1.014   217
sigma2[2,5]    0.012  0.015  -0.010   0.009   0.050     TRUE 0.852 1.011   218
sigma2[3,5]    0.009  0.024  -0.034   0.004   0.069     TRUE 0.714 1.013   316
sigma2[4,5]    0.026  0.016   0.003   0.023   0.064    FALSE 0.986 1.002 15000
sigma2[5,5]    0.056  0.029   0.021   0.049   0.127    FALSE 1.000 1.004   813
rho[1]         0.050  0.387  -0.670   0.056   0.760     TRUE 0.547 1.005   473
rho[2]         0.057  0.392  -0.707   0.066   0.762     TRUE 0.559 1.004   616
rho[3]         0.254  0.259  -0.289   0.269   0.710     TRUE 0.829 1.003   632
rho[4]         0.253  0.240  -0.226   0.263   0.699     TRUE 0.844 1.011   201
rho[5]         0.301  0.268  -0.245   0.318   0.760     TRUE 0.852 1.015   152
rho[6]         0.210  0.386  -0.612   0.247   0.836     TRUE 0.714 1.008   315
rho[7]         0.572  0.214   0.073   0.604   0.895    FALSE 0.986 1.006   683
rho[8]        -0.002  0.579  -0.949  -0.002   0.951     TRUE 0.501 1.000 15000
rho[9]        -0.007  0.577  -0.947  -0.019   0.950     TRUE 0.509 1.000  8712
rho[10]       -0.003  0.575  -0.950  -0.005   0.951     TRUE 0.502 1.000 15000
mean.p         0.795  0.033   0.728   0.796   0.856    FALSE 1.000 1.001  1845
mean.pj        0.607  0.039   0.530   0.607   0.683    FALSE 1.000 1.004   519
sigma.res      4.113  2.737   0.469   3.659  10.614    FALSE 1.000 1.005   398
phij[1]        0.125  0.025   0.080   0.123   0.178    FALSE 1.000 1.002  1045
phij[2]        0.145  0.026   0.100   0.143   0.201    FALSE 1.000 1.006   416
[...output truncated...]
phij[14]       0.093  0.021   0.054   0.093   0.133    FALSE 1.000 1.009   267
phij[15]       0.105  0.022   0.064   0.104   0.153    FALSE 1.000 1.003   762
phia[1]        0.371  0.043   0.282   0.369   0.461    FALSE 1.000 1.006   387
phia[2]        0.387  0.046   0.305   0.381   0.495    FALSE 1.000 1.005   436
[...output truncated...]
phia[14]       0.320  0.040   0.232   0.324   0.387    FALSE 1.000 1.006   373
phia[15]       0.392  0.043   0.324   0.384   0.496    FALSE 1.000 1.005   497
omega[1]       0.321  0.098   0.197   0.302   0.581    FALSE 1.000 1.011   427
omega[2]       0.347  0.116   0.194   0.322   0.674    FALSE 1.000 1.019   124
[...output truncated...]
omega[14]      0.262  0.069   0.117   0.263   0.398    FALSE 1.000 1.004  1157
omega[15]      0.291  0.068   0.170   0.287   0.438    FALSE 1.000 1.017   173
f1[1]          5.375  0.520   4.370   5.369   6.419    FALSE 1.000 1.001  5764
f1[2]          7.681  0.560   6.625   7.663   8.831    FALSE 1.000 1.005   866
[...output truncated...]
f1[14]         4.567  0.355   3.903   4.558   5.279    FALSE 1.000 1.001  8065
f1[15]         4.947  0.417   4.161   4.933   5.802    FALSE 1.000 1.002  1110
f2[1]          7.433  0.945   5.795   7.359   9.460    FALSE 1.000 1.012   186
f2[2]          6.896  0.677   5.606   6.885   8.258    FALSE 1.000 1.003   829
[...output truncated...]
f2[14]         4.015  0.289   3.473   4.007   4.586    FALSE 1.000 1.000  5661
f2[15]         5.626  0.413   4.861   5.613   6.459    FALSE 1.000 1.004   595
N[1,1]        12.118  8.669   1.000  10.000  32.000    FALSE 1.000 1.000 15000
N[2,1]        14.031  8.950   1.000  13.000  33.000    FALSE 1.000 1.000  5054
N[3,1]        11.999  8.674   1.000  10.000  32.000    FALSE 1.000 1.001  1692
```

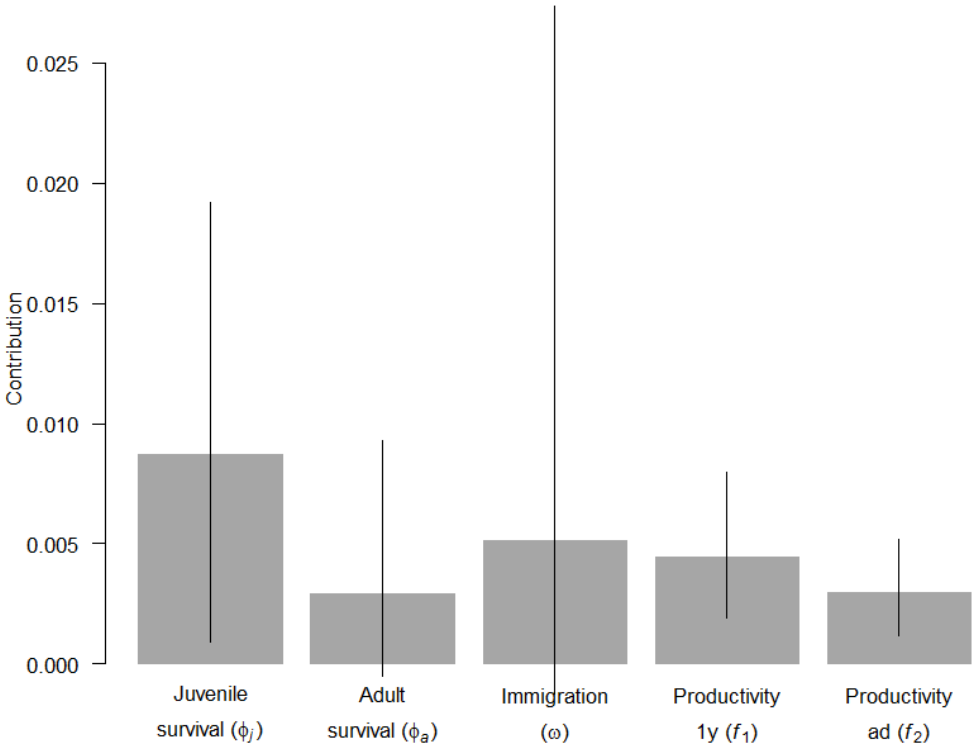
```
N[1,2]       17.030  4.615   8.000  17.000  26.000    FALSE 1.000 1.001  1665
N[2,2]       15.597  3.518   9.000  15.000  23.000    FALSE 1.000 1.005   465
N[3,2]       14.069  4.432   6.000  14.000  24.000    FALSE 1.000 1.002  1143
[...output truncated...]
N[1,15]      12.687  4.174   5.000  12.000  21.000    FALSE 1.000 1.005   439
N[2,15]      20.627  4.218  13.000  21.000  29.000    FALSE 1.000 1.003   764
N[3,15]      16.096  4.932   7.000  16.000  26.000    FALSE 1.000 1.002  1518
N[1,16]      13.714  4.203   6.000  14.000  22.000    FALSE 1.000 1.001  1446
N[2,16]      19.581  3.709  13.000  19.000  27.000    FALSE 1.000 1.001  2128
N[3,16]      14.470  4.355   6.000  14.000  23.000    FALSE 1.000 1.001  1752
Ntot[1]      38.148  5.229  32.000  37.000  52.000    FALSE 1.000 1.003   767
Ntot[2]      46.696  3.704  40.000  46.000  55.000    FALSE 1.000 1.001 15000
[...output truncated...]
Ntot[15]     49.410  3.754  43.000  49.000  58.000    FALSE 1.000 1.001  2796
Ntot[16]     47.765  4.125  39.000  48.000  56.000    FALSE 1.000 1.002  1586
lambda[1]     1.240  0.145   0.930   1.257   1.485    FALSE 1.000 1.003   610
lambda[2]     1.436  0.124   1.158   1.449   1.674    FALSE 1.000 1.001  4758
[...output truncated...]
lambda[14]    0.723  0.064   0.615   0.710   0.877    FALSE 1.000 1.002  1643
lambda[15]    0.970  0.090   0.780   0.979   1.149    FALSE 1.000 1.003   680
deviance    619.477 27.082 556.680 623.402 663.430    FALSE 1.000 1.011   256
```

After we have inspected the estimates and convinced ourselves that the MCMC chains appear to have converged, we will use these estimates to perform a classical LTRE. This comes next.

```
# Perform LTRE
# Calculation of growth rate sensitivities
delta <- 0.001                          # (Small) size of perturbation
T <- 5                                  # Number of years with projections
n.draws <- out1$mcmc.info$n.samples # Number of MCMC draws

# Define arrays to store output
N.ref <- N.star.phij <- N.star.phia <- N.star.om <- N.star.f1 <-
N.star.f2 <- array(NA, dim=c(n.draws, 3, T))
N.ref[,,1] <- N.star.phij[,,1] <- N.star.phia[,,1] <- N.star.om[,,1] <-
N.star.f1[,,1] <- N.star.f2[,,1] <- 1
r.annual.ref <- r.annual.star.phij <- r.annual.star.phia <- r.annual.star.om <-
r.annual.star.f1 <- r.annual.star.f2 <-
matrix(NA, nrow=n.draws, ncol=T)
lambda <- numeric()
sens <- matrix(NA, nrow=n.draws, ncol=5)
draws <- out1$sims.list   # store MCMC draws in a new object to simplify
calculation

# Loop over all MCMC draws and project in time
for (s in 1:n.draws){                    # Loop over all MCMC draws
  for (t in 1:(T-1)){                    # Loop over all time steps
    # Calculate asymptotic population growth rate based on stage-specific
population sizes
    N.ref[s,1,t+1] <- draws$mean.phij[s] * (draws$mean.f1[s] * (N.ref[s,1,t] +
N.ref[s,3,t]) + draws$mean.f2[s] * N.ref[s,2,t]) / 2
    N.ref[s,2,t+1] <- draws$mean.phia[s] * sum(N.ref[s,,t])
    N.ref[s,3,t+1] <- draws$mean.omega[s] * sum(N.ref[s,,t])
    r.annual.ref[s,t] <- log(sum(N.ref[s,,t+1])) - log(sum(N.ref[s,,t]))

    # Sensitivity with respect to juvenile survival
    N.star.phij[s,1,t+1] <- (draws$mean.phij[s] + delta) / 2 *
(draws$mean.f1[s] * (N.star.phij[s,1,t] + N.star.phij[s,3,t]) +
draws$mean.f2[s] * N.star.phij[s,2,t])
    N.star.phij[s,2,t+1] <- draws$mean.phia[s] * sum(N.star.phij[s,,t])
    N.star.phij[s,3,t+1] <- draws$mean.omega[s] * sum(N.star.phij[s,,t])
```

```
    r.annual.star.phij[s,t] <- log(sum(N.star.phij[s,,t+1])) -
log(sum(N.star.phij[s,,t]))

    # Sensitivity with respect to adult survival
    N.star.phia[s,1,t+1] <- draws$mean.phij[s] / 2 * (draws$mean.f1[s] *
(N.star.phia[s,1,t] + N.star.phia[s,3,t]) + draws$mean.f2[s] *
N.star.phia[s,2,t])
    N.star.phia[s,2,t+1] <- (draws$mean.phia[s] + delta) *
sum(N.star.phia[s,,t])
    N.star.phia[s,3,t+1] <- draws$mean.omega[s] * sum(N.star.phia[s,,t])
    r.annual.star.phia[s,t] <- log(sum(N.star.phia[s,,t+1])) -
log(sum(N.star.phia[s,,t]))

    # Sensitivity with respect to immigration
    N.star.om[s,1,t+1] <- draws$mean.phij[s] / 2 * (draws$mean.f1[s] *
(N.star.om[s,1,t] + N.star.om[s,3,t]) + draws$mean.f2[s] * N.star.om[s,2,t])
    N.star.om[s,2,t+1] <- draws$mean.phia[s] * sum(N.star.om[s,,t])
    N.star.om[s,3,t+1] <- (draws$mean.omega[s] + delta) * sum(N.star.om[s,,t])
    r.annual.star.om[s,t] <- log(sum(N.star.om[s,,t+1])) -
log(sum(N.star.om[s,,t]))

    # Sensitivity with respect to productivity 1y
    N.star.f1[s,1,t+1] <- draws$mean.phij[s] / 2 * ((draws$mean.f1[s] + delta)
* (N.star.f1[s,1,t] + N.star.f1[s,3,t]) + draws$mean.f2[s] * N.star.f1[s,2,t])
    N.star.f1[s,2,t+1] <- draws$mean.phia[s] * sum(N.star.f1[s,,t])
    N.star.f1[s,3,t+1] <- draws$mean.omega[s] * sum(N.star.f1[s,,t])
    r.annual.star.f1[s,t] <- log(sum(N.star.f1[s,,t+1])) -
log(sum(N.star.f1[s,,t]))

    # Sensitivity with respect to productivity ad
    N.star.f2[s,1,t+1] <- draws$mean.phij[s] / 2 * (draws$mean.f1[s] *
(N.star.f2[s,1,t] + N.star.f2[s,3,t]) + (draws$mean.f2[s]  + delta) *
N.star.f2[s,2,t])
    N.star.f2[s,2,t+1] <- draws$mean.phia[s] * sum(N.star.f2[s,,t])
    N.star.f2[s,3,t+1] <- draws$mean.omega[s] * sum(N.star.f2[s,,t])
    r.annual.star.f2[s,t] <- log(sum(N.star.f2[s,,t+1])) -
log(sum(N.star.f2[s,,t]))
  } #t

  lambda[s] <- exp(r.annual.ref[s,T-1])

  # Growth rate sensitivities
  sens[s,1] <- (exp(r.annual.star.phij[s,T-1]) - lambda[s]) / delta
  sens[s,2] <- (exp(r.annual.star.phia[s,T-1]) - lambda[s]) / delta
  sens[s,3] <- (exp(r.annual.star.om[s,T-1]) - lambda[s]) / delta
  sens[s,4] <- (exp(r.annual.star.f1[s,T-1]) - lambda[s]) / delta
  sens[s,5] <- (exp(r.annual.star.f2[s,T-1]) - lambda[s]) / delta
} #s


# Calculate variance-covariance matrix of the demographic rates
vcov <- array(NA, dim=c(n.draws, 5, 5))
for (s in 1:n.draws){
  vcov[s,,] <- var(cbind(draws$phij[s,], draws$phia[s,], draws$omega[s,],
draws$f1[s,1:15], draws$f2[s,1:15]))
}

# Calculate contribution matrix for every posterior draw
cont <- array(NA, dim=c(n.draws, 5, 5))
for (s in 1:n.draws){
  cont[s,,] <- vcov[s,,] * outer(sens[s,], sens[s,])
}
```

```
# Produce figure (same as Fig. 9.3 in the book)
CRI <- apply(apply(cont, c(1,2), sum), 2, quantile, probs=c(0.025, 0.975))
par(cex=1.1)
names.arg=c(expression(atop("Juvenile", "survival ("*phi[italic(j)]*")")),
            expression(atop("Adult", "survival ("*phi[italic(a)]*")")),
            expression(atop("Immigration", "("*omega*")")),
            expression(atop("Productivity", "1y ("*italic(f)[1]*")")),
            expression(atop("Productivity", "ad ("*italic(f)[2]*")")))
a <- barplot(apply(apply(cont, c(1,2), sum), 2, mean), ylim=range(CRI),
names.arg=names.arg, ylab="Contribution", xlab="", las=1, col="grey65",
border="grey65")
segments(a, CRI[1,], a, CRI[2,], lwd=1.5)
```

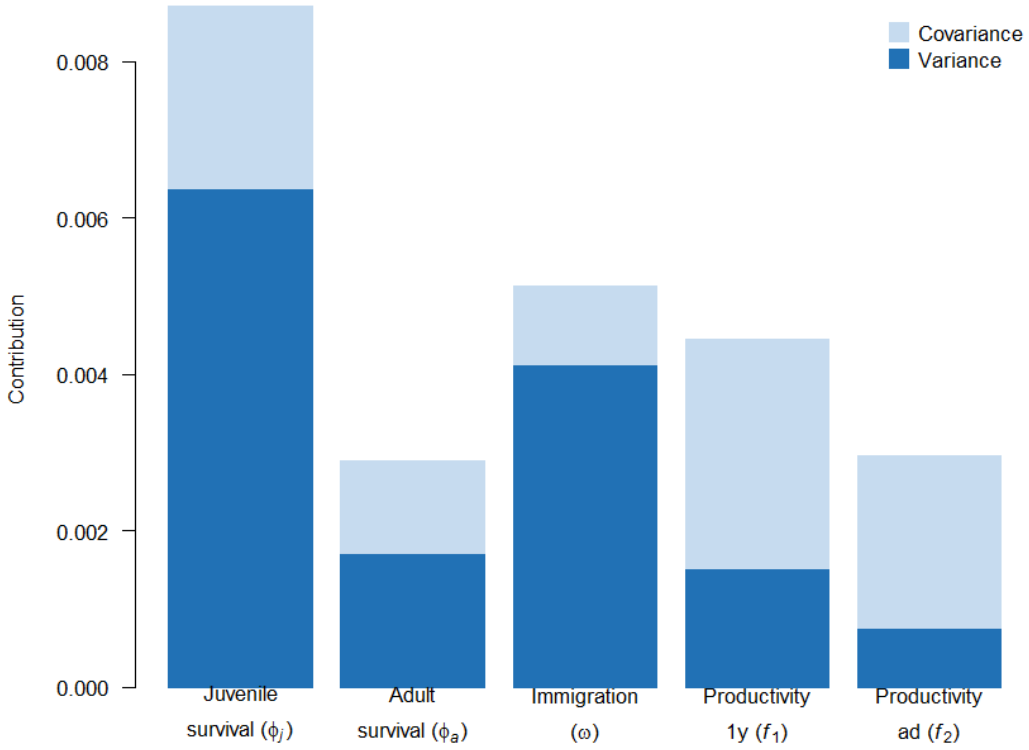


Here we see the estimates (posterior means with 95% CRIs) of the contributions of the realized temporal variability in each demographic rate to the temporal variability of the asymptotic population growth rate (λ_1) under the LTRE approach when applied to the Swiss hoopoe data (and remember: these are real hoopoes this time, not simulated ones). The results are very similar to those from the model without the correlations specified as parameters which we had seen in Section 9.3 (see Figure 9.3).

We can also determine how much a demographic rate has contributed through its own temporal variability and how much it has contributed via pairwise temporal covariation with the other demographic rates. We do this next.

```
mean.cont <- apply(cont, c(2, 3), mean)
mean.cont2 <- rbind(diag(mean.cont), rowSums(mean.cont) - diag(mean.cont))
```

```
library(RColorBrewer)
co <- brewer.pal(n=8, name='Blues')[c(7,3)]

par(mar=c(4, 5, 1, 1), cex=1.1)
names.arg=c(expression(atop("Juvenile", "survival ("*phi[italic(j)]*")")),
            expression(atop("Adult", "survival ("*phi[italic(a)]*")")),
            expression(atop("Immigration", "("*omega*")")),
            expression(atop("Productivity", "1y ("*italic(f)[1]*")")),
            expression(atop("Productivity", "ad ("*italic(f)[2]*")")))
a <- barplot(mean.cont2, names.arg=names.arg, ylab=NA, xlab=NA, col=co,
border=co, axes=FALSE)
axis(2, las=1)
mtext("Contribution", side=2, line=4)
legend("topright", legend=c("Covariance", "Variance"), col=co[2:1], pch=c(15,
15), bty="n", pt.cex=2)
```



Compared to the previous analysis that did not specify the temporal correlations as parameters in the model (see Figure 9.4), the magnitude of the contributions of the pairwise temporal correlations is now somewhat increased.

Exercise 2

The results of LTRE and tLTRE are typically associated with pronounced uncertainty. Sketch out a simulation study that aims to find out how strongly variable study duration and variable levels of temporal variability of demographic rates contributes to these uncertainties.

Solution. This is a rather substantial exercise that could easily become the content of a paper. We here only provide some pointers to important issues that need or should be considered when such a simulation study is performed. We do not claim to be exhaustive, and you may find other important aspects that we do not touch upon here. Here is a list of such ideas:

A. Decide on the dimension of the simulation. Shall we use a single or several life-histories (i.e., toy species)? Define typical values of the mean demographic rates for these toy species. We need to decide on how many levels of study durations we want to use (e.g., 10, 20, 30, 50 years) and about a gradient of the magnitude of temporal variability of the demographic rates. The latter needs particularly careful evaluation. Shall all demographic rates be temporally variable, or only a single one or a few? How large shall the different magnitudes of temporal variability be?
B. Decide about population size. The larger the sample size, the more precise the estimates will become in general and potentially we may be able to see more clearly the patterns in which we are primarily interested. Hence, we may decide to simulate data sets with many individuals (i.e., a large population). This has the additional advantage that the impact of demographic stochasticity is reduced, which would otherwise add more noise into our simulation study and might conceal the main patterns of interest. On the other hand, it would be interesting to also gauge the contribution of demographic stochasticity (which induces temporal variability in realized demographic rates).
C. Possible sketch of the simulation:
    a. Create a population given the life-history, demographic rates, their temporal variability, the population size and the study duration.
    b. Sample data sets from the population (e.g., counts, capture-recapture, productivity).
    c. Analyze the data in an IPM. Use temporal random effects for the demographic rates.
    d. Perform a tLTRE (or LTRE) using the results of the IPM.
    e. Repeat these steps several times (e.g., 10, 50 or 100 times) and store the results.
    f. Do such simulations for each scenario (different magnitude of temporal variability, different study durations).
D. Think about how the results should be presented. From the tLTRE we get the contribution of the temporal variability of each demographic rate (and population structure) to the variability of the population growth rate. What is our expectation? A rate that is not variable over time should not be identified as a population driver. If we want to assess the effects of increasing temporal variability in the demographic rate on the population contribution, it might be best to focus on a single parameter. The same is true if we want to demonstrate the impact of study duration. But what if we want to get the correct order of the demographic rates regarding their impact on population dynamics? How is the estimate of such an ordering affected by different magnitudes of temporal variability of different demographic rates? Perhaps it is also possible to calculate an expected value of the contribution of each demographic rate based on the simulation set up. If this is possible, then bias and MSE could be assessed.

The simulation functions in the `IPMbook` package should be quite useful to perform such simulation exercises (we would hope).

Chapter 10

Exercise 1

Let's assume that the woodchat shrike data from Section 10.5 refer in reality to a larger population and that in the last of our study years we surveyed the complete population and found that it consisted between 25 to 30 breeding pairs. The exercise is to assess the quasi-extinction probability (D = 5) within the next 15 years of the complete population.

Solution. The population count on which the IPM is based does not reflect the true population size, but instead is only an index of the true population size. Hence, the estimated 'true level' of the population (i.e., the latent N) is also an index. However, for extinction rate assessments, we need to work with the actual, 'true' population size, and not a mere index of it. Hence, to project the true population into the future, we must start from the 'true' population size, thus from the quantity we assume we obtained by the complete survey of the population in the last year (i.e., the 25–30 breeding pairs). Since there is uncertainty about the true size, we here use a uniform distribution with limits 25–30. For the population projection we also must allocate the individuals to the two age classes. We do this by using the estimated age distribution of the last study year. The remaining parts of the model are very similar to those of the model in Section 10.5. We load the woodchat shrike data, write the model, and run it.

```
# Load woodchat shrike data and produce data overview
library(IPMbook)
data(woodchat10)

# Bundle data
K <- 15                       # Number of years with predictions
jags.data <- list(marr.j=woodchat10$marr.j, marr.a=woodchat10$marr.a,
n.occasions=ncol(woodchat10$marr.j), rel.j=rowSums(woodchat10$marr.j),
rel.a=rowSums(woodchat10$marr.a), J=woodchat10$J, B=woodchat10$B,
count=woodchat10$count, pNinit=dUnif(1, 50), K=K)


# Write JAGS model file
cat(file="model1.txt", "
model {
# Priors and linear models
mean.logit.sj <- logit(mean.sj)
mean.sj ~ dunif(0, 1)
mean.logit.sa <- logit(mean.sa)
mean.sa ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.log.f <- log(mean.f)
mean.f ~ dunif(0, 10)

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

for (t in 1:(n.occasions-1+K)){  # Here we extend the loop to K more years
  logit.sj[t] <- mean.logit.sj + eps.sj[t]
  eps.sj[t] ~ dnorm(0, tau.sj)
  sj[t] <- ilogit(logit.sj[t])
  logit.sa[t] <- mean.logit.sa + eps.sa[t]
  eps.sa[t] ~ dnorm(0, tau.sa)
```

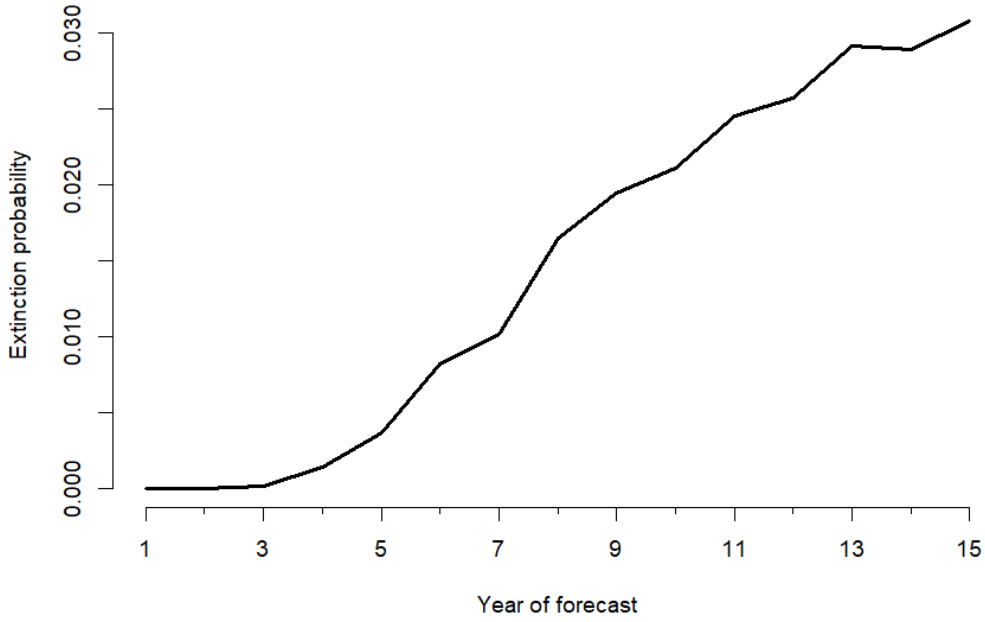
```
  sa[t] <- ilogit(logit.sa[t])
}

for (t in 1:(n.occasions+K)){   # Extended loop also here
  log.f[t] <- mean.log.f + eps.f[t]
  eps.f[t] ~ dnorm(0, tau.f)
  f[t] <- exp(log.f[t])
}

sigma.sj ~ dunif(0, 10)
tau.sj <- pow(sigma.sj, -2)
sigma.sa ~ dunif(0, 10)
tau.sa <- pow(sigma.sa, -2)
sigma.f ~ dunif(0, 10)
tau.f <- pow(sigma.f, -2)

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

# Population count data (state-space model)
# Model for the initial population size: uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

# Process model over time: our model of population dynamics
for (t in 1:(n.occasions-1)){
  N[1,t+1] ~ dpois(sj[t] * f[t] * (N[1,t] + N[2,t]))
  N[2,t+1] ~ dbin(sa[t], (N[1,t] + N[2,t]))
}

# Process model over time: prediction
Ntrue ~ dunif(25, 30)    # Uncertainty of population size
# Age ratio in the population in the last year
ageRatio[1] <- N[1,n.occasions] / (N[1,n.occasions] + N[2,n.occasions])
ageRatio[2] <- N[2,n.occasions] / (N[1,n.occasions] + N[2,n.occasions])
# Calculate age-specific 'true' population size in the last study year
Np[1,1] <- round(Ntrue * ageRatio[1])
Np[2,1] <- round(Ntrue * ageRatio[2])
# Population prediction
for (t in 1:K){
  Np[1,t+1] ~ dpois(sj[n.occasions-1+t] * f[n.occasions-1+t] * (Np[1,t] +
Np[2,t]))
  Np[2,t+1] ~ dbin(sa[n.occasions-1+t], (Np[1,t] + Np[2,t]))
}

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

# Productivity data (Poisson regression model)
for (t in 1:n.occasions){
  J[t] ~ dpois(f[t] * B[t])
}

# Capture-recapture data (CJS model with multinomial likelihood)
# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
  marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], rel.j[t])
  marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], rel.a[t])
}
# Define the cell probabilities of the m-arrays
```

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

# Derived parameters
# Total population size in the future
for (t in 1:K){
  Ntot[t] <- Np[1,t] + Np[2,t]
}
# Check whether the population is extinct in the future
for (t in 1:K){
  extinct[t] <- equals(Ntot[t]-5, 0)  # Extinction threshold = 5
}
}
")

# Initial values
# Give initial values for N, otherwise the MCMC may not work
N <- matrix(round(runif(40, 10, 20)), nrow=2, ncol=20)
inits <- function(){list(mean.sj=runif(1, 0, 0.5), mean.sa=runif(1, 0.4, 0.6),
mean.f=runif(1, 1.3, 2), N=N)}

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

# MCMC settings
ni <- 20000; nb <- 5000; nc <- 3; nt <- 3; na <- 1000

# Call JAGS (ART 2 min), check convergence and summarize posteriors
out1 <- jags(jags.data, inits, parameters, "model1.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out1)      # Not shown
print(out1, 3)       # Not shown
```

The model converges well. We next visualize the main results and plot the cumulative quasi-extinction probabilities.

```
# Produce plot to show quasi-extinction probabilities
par(cex=1.2)
plot(out1$mean$extinct, type="l", ylab="Extinction probability", lwd=3,
xlab="Year of forecast", frame=FALSE, axes=FALSE)
axis(1, at=1:K, tck=-0.0125, labels=FALSE)
```

```
axis(1, at=c(1, 3, 5, 7, 9, 11, 13, 15), labels=c(1, 3, 5, 7, 9, 11, 13, 15),
tck=-0.025)
axis(2)
```



The plot shows the cumulative quasi-extinction probability (with an extinction threshold of D = 5) when the simulated woodchat shrike population is forecast into the future and the population size in the last year of study (when we assumed complete coverage) was between 25 and 30 breeding pairs. The quasi-extinction probability first increases and then stabilizes. You can confirm this behavior by increasing the projection duration from 15 to, say, 30 years. This suggests that if extinction occurs, it is likely to happen within a relatively short time.

Exercise 2

Let's assume that we want to create a self-sustainable hoopoe population, i.e., a population that can survive without immigration. How much does juvenile survival need to increase such that the extinction risk of the hoopoe population becomes less than 0.1 in 10 years?

Solution. To solve this exercise, we first need to rewrite the population model such that it does not include immigration anymore. This requires changes at several places in the original code that you find in Section 10.6. To assess how much juvenile survival needs to be increased to obtain an extinction risk of less than 0.1 in 10 years, we here choose a brute-force approach. That is, we forecast the population under different scenarios, each of them representing a different magnitude of the hypothetical increase of juvenile survival. We then check for what value of juvenile survival we meet our objective. We choose 4 levels: 10, 20, 30 and 40% increase of juvenile survival and then will interpolate between these values to obtain our answer. If you

want to get a more precise result, then you simply need to add more levels. We do not calculate the extinction probability inside of the BUGS code, but do this calculation afterwards in R. We start by loading the hoopoe data, write the model, run it, and summarize the results.

```
# Load hoopoe data
library(IPMbook)
data(hoopoe)

# Produce m-array
marr <- marrayAge(hoopoe$ch, hoopoe$age)
marr.j <- marr[,,1]
marr.a <- marr[,,2]

# Bundle data
jags.data <- with(hoopoe$reproAgg, list(marr.j=marr.j, marr.a=marr.a,
n.occasions=ncol(marr.j), rel.j=rowSums(marr.j), rel.a=rowSums(marr.a), J1=J1,
B1=B1, J2=J2, B2=B2, count=hoopoe$count, pNinit=dUnif(1, 50), K=10))

# Write JAGS model file
cat(file="model2.txt", "
model {
# Priors and linear models
# For mean and variance parameters
mean.phij ~ dunif(0, 1)
mean.phia ~ dunif(0, 1)
mean.f1 ~ dunif(0, 10)
mean.f2 ~ dunif(0, 10)

mu[1] <- logit(mean.phij)
mu[2] <- logit(mean.phia)
mu[3] <- log(mean.f1)
mu[4] <- log(mean.f2)

mean.p ~ dunif(0, 1)
mean.pj ~ dunif(0, 1)

# Linear models for demographic parameters
for (t in 1:(n.occasions-1+K)){
  logit(phij[t]) <- eps[1,t]
  logit(phia[t]) <- eps[2,t]
  log(f1[t]) <- eps[3,t]
  log(f2[t]) <- eps[4,t]
  eps[1:4,t] ~ dmnorm.vcov(mu[], sigma2[,])
}

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

# Reparameterize sigma2 in terms of SD and rho
for (i in 1:4){
  sigma2[i,i] <- sigma[i] * sigma[i]
}
for (i in 1:3){
  for (j in (i+1):4){
    sigma2[i,j] <- sigma[i] * sigma[j] * rho[i,j]
    sigma2[j,i] <- sigma2[i,j]
    rho[i,j] ~ dunif(-1, 1)   # Prior for rho
```

```
  } #j
} #i

# Specify priors for SD
for (i in 1:4){
  sigma[i] ~ dunif(0, 5)
}

sigma.res ~ dunif(0.4, 20)
tau.res <- pow(sigma.res, -2)

# Population count data (state-space model)
# Model for initial stage-spec. population sizes: discrete uniform priors
N[1,1] ~ dcat(pNinit)
N[2,1] ~ dcat(pNinit)

# Process model over time: our model of population dynamics
for (t in 1:(n.occasions-1)){
  N[1,t+1] ~ dpois(f1[t] / 2 * phij[t] * N[1,t] + f2[t] / 2 * phij[t] * N[2,t])
  N[2,t+1] ~ dbin(phia[t], (N[1,t] + N[2,t]))
}

# Observation model
for (t in 1:n.occasions){
  count[t] ~ dnorm(Ntot[t], tau.res)
}
for (t in 1:(n.occasions)){
  Ntot[t] <- N[1,t] + N[2,t]         # total population size
}

# Predictions using different scenarios for increase of juvenile survival
# Scenario 1: increase by 10%
Np[1,1,1] <- N[1,n.occasions]
Np[1,2,1] <- N[2,n.occasions]
for (t in 1:K){
  phijp[1,t] <- phij[n.occasions-1+t] * 1.1
  Np[1,1,t+1] ~ dpois(f1[n.occasions-1+t] / 2 * phijp[1,t] * Np[1,1,t] +
f2[n.occasions-1+t] / 2 * phijp[1,t] * Np[1,2,t])
  Np[1,2,t+1] ~ dbin(phia[n.occasions-1+t], (Np[1,1,t] + Np[1,2,t]))
  Ntotp[1,t] <- Np[1,1,t+1] + Np[1,2,t+1]
}

# Scenario 2: increase by 20%
Np[2,1,1] <- N[1,n.occasions]
Np[2,2,1] <- N[2,n.occasions]
for (t in 1:K){
  phijp[2,t] <- phij[n.occasions-1+t] * 1.2
  Np[2,1,t+1] ~ dpois(f1[n.occasions-1+t] / 2 * phijp[2,t] * Np[2,1,t] +
f2[n.occasions-1+t] / 2 * phijp[2,t] * Np[2,2,t])
  Np[2,2,t+1] ~ dbin(phia[n.occasions-1+t], (Np[2,1,t] + Np[2,2,t]))
  Ntotp[2,t] <- Np[2,1,t+1] + Np[2,2,t+1]
}

# Scenario 3: increase by 30%
Np[3,1,1] <- N[1,n.occasions]
Np[3,2,1] <- N[2,n.occasions]
for (t in 1:K){
  phijp[3,t] <- phij[n.occasions-1+t] * 1.3
  Np[3,1,t+1] ~ dpois(f1[n.occasions-1+t] / 2 * phijp[3,t] * Np[3,1,t] +
f2[n.occasions-1+t] / 2 * phijp[3,t] * Np[3,2,t])
  Np[3,2,t+1] ~ dbin(phia[n.occasions-1+t], (Np[3,1,t] + Np[3,2,t]))
  Ntotp[3,t] <- Np[3,1,t+1] + Np[3,2,t+1]
```

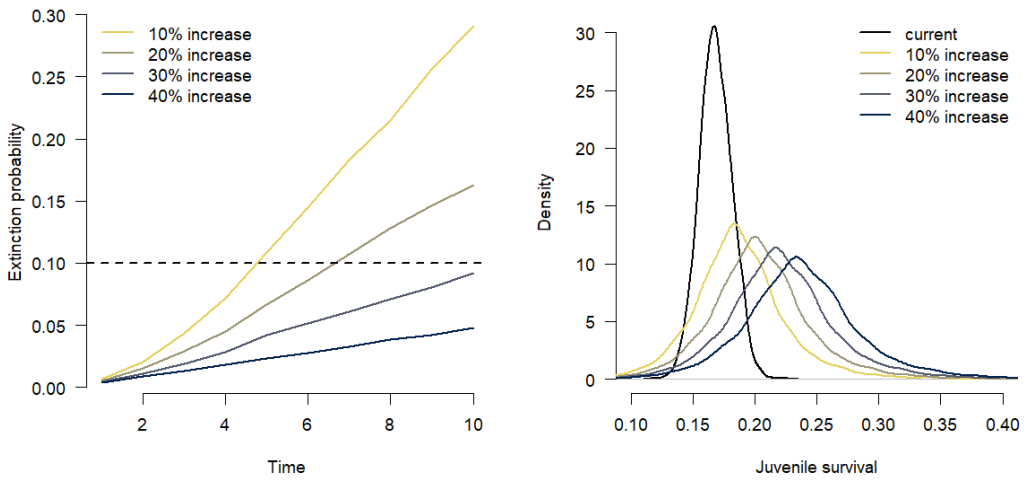
```
}

# Scenario 4: increase by 40%
Np[4,1,1] <- N[1,n.occasions]
Np[4,2,1] <- N[2,n.occasions]
for (t in 1:K){
  phijp[4,t] <- phij[n.occasions-1+t] * 1.4
  Np[4,1,t+1] ~ dpois(f1[n.occasions-1+t] / 2 * phijp[4,t] * Np[4,1,t] +
f2[n.occasions-1+t] / 2 * phijp[4,t] * Np[4,2,t])
  Np[4,2,t+1] ~ dbin(phia[n.occasions-1+t], (Np[4,1,t] + Np[4,2,t]))
  Ntotp[4,t] <- Np[4,1,t+1] + Np[4,2,t+1]
}

# Productivity data (Poisson regression model)
for (t in 1:(n.occasions-1)){
  J1[t] ~ dpois(f1[t] * B1[t])
  J2[t] ~ dpois(f2[t] * B2[t])
}

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

# Initial values
inits <- function(){list(mean.phij=runif(1, 0.05, 0.2), mean.phia=runif(1, 0.3,
0.5))}

# Parameters monitored
parameters <- c("mean.phij", "mean.phia", "mean.f1", "mean.f2", "mean.p",
"mean.pj", "sigma.res", "rho", "sigma2", "phij", "phia", "f1", "f2", "N",
"Ntot", "Np", "Ntotp", "phijp")

# MCMC settings
```

```
ni <- 30000; nb <- 10000; nc <- 3; nt <- 5; na <- 2000

# Call JAGS (ART 7 min), check convergence and summarize posteriors
out2 <- jags(jags.data, inits, parameters, "model2.txt", n.iter=ni,
n.burnin=nb, n.chains=nc, n.thin=nt, n.adapt=na, parallel=TRUE)
traceplot(out2)    # Not shown
print(out2)

JAGS output for model 'model2.txt', generated by jagsUI.
Estimates based on 3 chains of 30000 iterations,
adaptation = 2000 iterations (sufficient),
burn-in = 10000 iterations and thin rate = 5,
yielding 12000 total samples from the joint posterior.
MCMC ran in parallel for 7.488 minutes at time 2021-10-15 10:43:13.

               mean      sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
mean.phij     0.169   0.013   0.143   0.168   0.195    FALSE 1.000 1.003  1062
mean.phia     0.387   0.022   0.346   0.386   0.432    FALSE 1.000 1.006   630
mean.f1       5.579   0.309   4.986   5.571   6.214    FALSE 1.000 1.001  2410
mean.f2       6.228   0.415   5.463   6.202   7.121    FALSE 1.000 1.002  1423
mean.p        0.738   0.037   0.663   0.739   0.809    FALSE 1.000 1.001  2932
mean.pj       0.502   0.038   0.429   0.502   0.579    FALSE 1.000 1.001  1784
sigma.res     7.918   4.544   1.041   7.192  18.346    FALSE 1.000 1.002   843
rho[1,2]      0.006   0.431  -0.777  -0.001   0.828     TRUE 0.499 1.005  2477
rho[1,3]      0.300   0.346  -0.455   0.336   0.870     TRUE 0.804 1.009   227
rho[2,3]      0.171   0.388  -0.643   0.197   0.815     TRUE 0.678 1.002   856
rho[1,4]      0.233   0.347  -0.516   0.263   0.815     TRUE 0.753 1.010   220
rho[2,4]      0.364   0.362  -0.469   0.418   0.909     TRUE 0.835 1.001  2176
rho[3,4]      0.569   0.225   0.039   0.606   0.896    FALSE 0.982 1.009   270
sigma2[1,1]   0.055   0.053   0.000   0.042   0.190    FALSE 1.000 1.007   808
sigma2[2,1]   0.000   0.019  -0.040   0.000   0.040     TRUE 0.501 1.015 12000
sigma2[3,1]   0.013   0.019  -0.016   0.011   0.055     TRUE 0.804 1.003   659
sigma2[4,1]   0.013   0.021  -0.024   0.010   0.063     TRUE 0.753 1.004   655
sigma2[1,2]   0.000   0.019  -0.040   0.000   0.040     TRUE 0.501 1.015 12000
sigma2[2,2]   0.037   0.043   0.000   0.025   0.153    FALSE 1.000 1.020   357
sigma2[3,2]   0.007   0.016  -0.021   0.004   0.043     TRUE 0.678 1.005  2870
sigma2[4,2]   0.016   0.020  -0.012   0.012   0.066     TRUE 0.835 1.004  4043
sigma2[1,3]   0.013   0.019  -0.016   0.011   0.055     TRUE 0.804 1.003   659
sigma2[2,3]   0.007   0.016  -0.021   0.004   0.043     TRUE 0.678 1.005  2870
sigma2[3,3]   0.041   0.021   0.016   0.036   0.095    FALSE 1.000 1.001 12000
sigma2[4,3]   0.027   0.017   0.002   0.024   0.070    FALSE 0.982 1.002   965
sigma2[1,4]   0.013   0.021  -0.024   0.010   0.063     TRUE 0.753 1.004   655
sigma2[2,4]   0.016   0.020  -0.012   0.012   0.066     TRUE 0.835 1.004  4043
sigma2[3,4]   0.027   0.017   0.002   0.024   0.070    FALSE 0.982 1.002   965
sigma2[4,4]   0.058   0.031   0.022   0.051   0.137    FALSE 1.000 1.002  4077
phij[1]       0.169   0.026   0.119   0.168   0.222    FALSE 1.000 1.004   827
phij[2]       0.191   0.028   0.143   0.189   0.251    FALSE 1.000 1.009   262
[...output truncated...]
phij[24]      0.171   0.036   0.107   0.169   0.251    FALSE 1.000 1.001  3492
phij[25]      0.171   0.035   0.105   0.169   0.249    FALSE 1.000 1.000  5915
phia[1]       0.400   0.040   0.325   0.396   0.489    FALSE 1.000 1.012   578
phia[2]       0.401   0.044   0.324   0.396   0.500    FALSE 1.000 1.018   229
[...output truncated...]
phia[24]      0.387   0.050   0.289   0.384   0.498    FALSE 1.000 1.004  2024
phia[25]      0.389   0.050   0.294   0.385   0.504    FALSE 1.000 1.002  2144
f1[1]         5.399   0.516   4.423   5.383   6.444    FALSE 1.000 1.000  9593
f1[2]         7.781   0.568   6.742   7.761   8.931    FALSE 1.000 1.001  3522
[...output truncated...]
f1[24]        5.685   1.219   3.669   5.572   8.472    FALSE 1.000 1.001 10929
f1[25]        5.699   1.234   3.661   5.576   8.516    FALSE 1.000 1.000 12000
f2[1]         7.681   0.956   6.020   7.608   9.739    FALSE 1.000 1.006   328
f2[2]         7.035   0.693   5.751   7.014   8.444    FALSE 1.000 1.001  2347
[...output truncated...]
f2[24]        6.400   1.677   3.778   6.181  10.230    FALSE 1.000 1.000  9385
f2[25]        6.445   1.691   3.837   6.233  10.499    FALSE 1.000 1.000 12000
N[1,1]       17.869  12.838   1.000  15.000  46.000    FALSE 1.000 1.000  9157
N[2,1]       27.526  11.361   4.000  29.000  47.000    FALSE 1.000 1.000  3711
N[1,2]       30.871   5.589  21.000  31.000  43.000    FALSE 1.000 1.001  4157
```

```
N[2,2]       19.989   4.825  12.000  20.000  30.000    FALSE 1.000 1.002  2010
[...output truncated...]
N[1,15]      21.721   5.782  10.000  22.000  33.000    FALSE 1.000 1.001  4788
N[2,15]      23.150   5.057  13.000  23.000  33.000    FALSE 1.000 1.002  1078
N[1,16]      20.962   6.148   8.000  21.000  32.000    FALSE 1.000 1.002   868
N[2,16]      19.494   4.709  10.000  20.000  28.000    FALSE 1.000 1.001  2332
Ntot[1]      45.395   9.324  33.000  44.000  67.000    FALSE 1.000 1.002   978
Ntot[2]      50.860   6.526  41.000  50.000  67.000    FALSE 1.000 1.000 12000
[...output truncated...]
Ntot[15]     44.871   7.377  26.000  47.000  56.000    FALSE 1.000 1.001  1386
Ntot[16]     40.456   8.421  20.000  42.000  52.000    FALSE 1.000 1.002  1163
Np[1,1,1]    20.962   6.148   8.000  21.000  32.000    FALSE 1.000 1.002   868
Np[2,1,1]    20.962   6.148   8.000  21.000  32.000    FALSE 1.000 1.002   868
Np[3,1,1]    20.962   6.148   8.000  21.000  32.000    FALSE 1.000 1.002   868
Np[4,1,1]    20.962   6.148   8.000  21.000  32.000    FALSE 1.000 1.002   868
Np[1,2,1]    19.494   4.709  10.000  20.000  28.000    FALSE 1.000 1.001  2332
Np[2,2,1]    19.494   4.709  10.000  20.000  28.000    FALSE 1.000 1.001  2332
Np[3,2,1]    19.494   4.709  10.000  20.000  28.000    FALSE 1.000 1.001  2332
Np[4,2,1]    19.494   4.709  10.000  20.000  28.000    FALSE 1.000 1.001  2332
 [...output truncated...]
Np[1,1,11]   19.083  30.642   0.000  11.000  83.000     TRUE 1.000 1.012  4413
Np[2,1,11]   33.259  54.670   0.000  20.000 144.000     TRUE 1.000 1.013  4607
Np[3,1,11]   56.988  98.869   1.000  34.000 250.000    FALSE 1.000 1.014  5686
Np[4,1,11]   92.926 167.775   3.000  54.000 403.000    FALSE 1.000 1.023  7512
Np[1,2,11]   12.408  17.385   0.000   8.000  50.000     TRUE 1.000 1.011  3933
Np[2,2,11]   19.816  27.119   0.000  13.000  79.025     TRUE 1.000 1.008  3076
Np[3,2,11]   31.241  44.304   1.000  21.000 123.025    FALSE 1.000 1.009  3292
Np[4,2,11]   47.671  72.706   2.000  31.000 187.000    FALSE 1.000 1.018  4551
Ntotp[1,1]   39.081  13.344  15.000  38.000  68.000    FALSE 1.000 1.000  4374
Ntotp[2,1]   41.169  14.171  16.000  40.000  72.000    FALSE 1.000 1.000  6423
Ntotp[3,1]   43.401  15.108  17.000  43.000  76.000    FALSE 1.000 1.000  4643
Ntotp[4,1]   45.518  15.832  17.000  45.000  79.000    FALSE 1.000 1.001  4825
 [...output truncated...]
Ntotp[1,10]  31.491  46.930   0.000  20.000 132.000     TRUE 1.000 1.013  4069
Ntotp[2,10]  53.075  80.043   1.000  33.000 223.000    FALSE 1.000 1.011  3825
Ntotp[3,10]  88.229 140.639   3.000  55.000 367.025    FALSE 1.000 1.013  4578
Ntotp[4,10] 140.597 236.590   6.000  85.000 577.025    FALSE 1.000 1.022  6165
phijp[1,1]    0.189   0.040   0.116   0.186   0.277    FALSE 1.000 1.001  4522
phijp[2,1]    0.206   0.043   0.127   0.203   0.302    FALSE 1.000 1.001  4522
phijp[3,1]    0.223   0.047   0.138   0.220   0.327    FALSE 1.000 1.001  4522
phijp[4,1]    0.240   0.051   0.148   0.237   0.352    FALSE 1.000 1.001  4522
 [...output truncated...]
phijp[1,10]   0.188   0.039   0.116   0.185   0.274    FALSE 1.000 1.000  5915
phijp[2,10]   0.205   0.042   0.127   0.202   0.299    FALSE 1.000 1.000  5915
phijp[3,10]   0.222   0.046   0.137   0.219   0.324    FALSE 1.000 1.000  5915
phijp[4,10]   0.239   0.049   0.148   0.236   0.349    FALSE 1.000 1.000  5915
deviance    674.442  23.248 616.619 677.502 711.428    FALSE 1.000 1.002  2123
```

```
# Calculate cumulative extinction probabilities
D <- 10       # Define extinction threshold
ext.prob <- matrix(NA, nrow=4, ncol=10)
extinct <- (out2$sims.list$Ntotp[,1,] <= D)
ext.prob[1,] <- apply(extinct, 2, mean)
extinct <- (out2$sims.list$Ntotp[,2,] <= D)
ext.prob[2,] <- apply(extinct, 2, mean)
extinct <- (out2$sims.list$Ntotp[,3,] <= D)
ext.prob[3,] <- apply(extinct, 2, mean)
extinct <- (out2$sims.list$Ntotp[,4,] <= D)
ext.prob[4,] <- apply(extinct, 2, mean)

# Produce plots
library(scales)
color <- viridis_pal(option='E')(20)[c(18,13,7,1)]
par(mfrow=c(1, 2), cex=1.2)
plot(ext.prob[1,], type="l", ylab="Extinction probability", xlab="Time",
axes=FALSE, col=color[1], lwd=2.5)
```

```
axis(1); axis(2, las=1)
lines(ext.prob[2,], col=color[2], lwd=2.5)
lines(ext.prob[3,], col=color[3], lwd=2.5)
lines(ext.prob[4,], col=color[4], lwd=2.5)
legend("topleft", legend=c("10% increase", "20% increase", "30% increase", "40%
increase"), col=color, lwd=2.5, bty="n")
abline(h=0.1, lwd=2, lty=2)

plot(density(out2$sims.list$mean.phij), type="l", ylab="Density",
xlab="Juvenile survival",  axes=FALSE, col="black", lwd=2.5, xlim=c(0.1, 0.4),
main=NA)
axis(1); axis(2, las=1)
lines(density(out2$sims.list$phijp[,1,10]), col=color[1], lwd=2.5)
lines(density(out2$sims.list$phijp[,2,10]), col=color[2], lwd=2.5)
lines(density(out2$sims.list$phijp[,3,10]), col=color[3], lwd=2.5)
lines(density(out2$sims.list$phijp[,4,10]), col=color[4], lwd=2.5)
legend("topright", legend=c("current", "10% increase", "20% increase", "30%
increase", "40% increase"), col=c("black", color), lwd=2.5, bty="n")
```



The first plot shows the cumulate quasi-extinction probabilities for the four levels of increased juvenile survival probabilities chosen. The brute-force approach suggests that survival needs to be increased by about 25% to achieve the stated goal of an extinction risk <0.1 in 10 years (we get this by visually interpolating the two middle lines in year 10 in the plot on the left). If you want to have a more precise result, you could repeat the analysis, but this time now vary the levels of increase only around 25% (e.g., 23, 24, 25, 26 and 27%). The second plot shows the posterior distributions of the current and the increased juvenile survival probabilities. The required increase of 25% corresponds to a mean value of juvenile survival of about 0.21.