CHAPTER

7

Estimation of Survival from Capture–Recapture Data Using the Cormack–Jolly–Seber Model

OUTLINE

Bayesian Population Analysis using WinBUGS
DOI: 10.1016/B978-0-12-387020-9.00007-9

7.1 INTRODUCTION

The preceding chapters dealt with the modeling and estimation of population size and with the simplest summary of population dynamics: population trend. The following chapters focus on one of the main components of population dynamics: survival probability. Survival probability is a key demographic parameter and can have a strong impact on population dynamics (Clobert and Lebreton, 1991; Saether and Bakke, 2000). Typically, interest focuses on estimation (what is the survival in that population?) as well as on modeling, for example, to test whether survival changes with age or differs between groups of individuals or regions, and to estimate how strongly it varies over time or what proportion of temporal variability can be explained by an external covariate such as weather.

In principle, survival estimation is fairly simple—we just have to count the number of individuals alive at a given time t (C_t), and keep track of how many of them die ($D_{\Delta t}$) during the period Δt for which we wish to estimate survival. Sometimes, it may be easier to count the number of the C_t that are still alive at $t + \Delta t$, that is, the number that survived the period Δt ($L_{\Delta t}$). Then survival probability s_t is

$$s_t = \frac{C_t - D_{\Delta t}}{C_t} = \frac{L_{\Delta t}}{C_t}$$

These numbers may easily be obtained in humans, but they are difficult to get in animal or plant populations. The reason for that is because the detection of individuals is usually far from perfect, so when an individual is not seen, we don't know whether it is dead or still alive. Therefore, such simple calculations cannot often be used, and we need to account for the

observation process in our inferences about survival (an exception being data on individuals with radio tags, White and Garrott, 1990). From the number of individuals recorded at t (C_t), we typically detect only a fraction of those still alive at time $t + \Delta t$, which is $p^* L_{\Delta t}$: p is the recapture or resighting probability (depending on the context or study design), which needs to be estimated in order to obtain unbiased estimates of survival. Estimating p becomes possible if we extend the recapture study to at least one further time step ($t + 2\Delta t$). We may then have individuals that are known to have survived until $t + 2\Delta t$, but which have not been seen at time $t + \Delta t$. Intuitively, it is clear that the proportion of these individuals provides information on p.

The most common statistical method to jointly estimate recapture and survival probabilities in animal and plant populations is a class of open population capture–recapture models to which the Cormack–Jolly–Seber (CJS) model belongs (Cormack, 1964; Jolly, 1965; Seber, 1965). Re-encounters may be obtained by different methods (physical capture, sightings, genetic tracking), but the key is that individuals are identified without error. That is, we only have false negatives, but no false positives. The frequentist analysis of the CJS model is described in detail in Lebreton et al. (1992) and Williams et al. (2002). Descriptions and examples of the Bayesian analysis of the CJS model can be found in an increasing number of articles and books (Brooks et al., 2000a; McCarthy and Masters, 2005; Gimenez et al., 2007; McCarthy, 2007; Zheng et al., 2007; Royle, 2008; Royle and Dorazio, 2008; Schofield et al., 2009; Gimenez et al., 2009a; King et al., 2010).

The CJS model can be fitted using either a multinomial (Lebreton et al., 1992) or a state-space likelihood (Gimenez et al., 2007; Royle, 2008). Because these two likelihoods are just different ways of describing what is essentially the same model, they are based on the same sampling design and the same underlying model assumptions. The sampling design is as follows: A random sample of individuals from the study population is captured, all are marked individually, and released into the population again. This is repeated several times. The length of the time intervals between repeated capture occasions depends on the research question, as well as on the life history and population dynamics of the study organism, and capture should be instantaneous or over a short time period. Some marked individuals will be re-encountered, and thus, we obtain capture–recapture data that can be summarized in individual capture-histories.

The CJS model makes a number of assumptions and, as usual, their violation may bias parameter estimators. Some assumptions must be met at the design stage of a study. Tags or other marks must not be lost, otherwise survival is underestimated. If mark loss is suspected, double marking and corresponding model adaptation that account for mark loss are necessary to get unbiased estimates of survival (e.g., Smout et al., 2011).

Ideally, capture should be instantaneous, otherwise the interval between capture occasions may differ among individuals and, consequently, there will be individual heterogeneity in survival. However, simulation studies have suggested that the violation of this last assumption does often not have a strong effect on parameters estimates (Hargrove and Borland, 1994). The CJS model also assumes that the identity of the individuals is always recorded without errors. If this assumption is violated, bias can go in either direction, and there is no means to correct for it. Finally, captured and recaptured individuals are regarded as a random sample from the study population. This sounds easy, but in practice, it can be difficult to achieve. For example, in studies on birds using nest boxes, it is quite typical that adults are only captured after the young have hatched because they are likely to abandon their brood if they are disturbed at an early stage. This results in a sample that is biased toward successfully breeding adults, which may or may not be a random sample from all adults in the population.

Further assumptions of the CJS model cannot be violated or fulfilled by the design of the study, rather they are a consequence of how the model is specified. The basic model assumes that each individual within an age class or group has the same survival and recapture probability. Goodness-of-fit tests help to identify severe violation of these assumptions (e.g., trap-response: Pradel, 1993; transients: Pradel et al., 1997), and modifications to the model allow to account for these violations. Individuals must behave independently from each other in terms of survival and recapture. This may not be the case if members of the same family are included in a sample and especially if they remain in family groups. Violation of this assumption is like pseudo-replication (Hurlbert, 1984). The degree of nonindependence leading to overdispersion can be estimated and the standard errors of the estimates as well as AIC-based model selection can be adjusted accordingly (Anderson et al., 1994).

With CJS models, we estimate recapture probability (p_t; the probability of catching/resighting a marked individual at t that is alive and in the sampling population at t) and apparent (also called "local") survival probability (ϕ_t; the probability that an individual that is alive and in the population at t is still alive and in the population at time $t+1$). Mortality and permanent emigration are confounded, and therefore apparent survival is always lower than true survival whenever permanent emigration is not zero. The difference between apparent and true survival is a matter of study design. Generally, the larger a study area the closer the match between apparent and true survival because dispersing individuals have a higher probability to remain in the study area (Marshall et al., 2004). Throughout this chapter, we will often just write "survival" for ease of presentation—but it is important to remember that survival in the CJS model always refers to a study area.

In this chapter, we introduce the CJS model and illustrate how it is fitted using the state-space (Sections 7.2–7.8) and the multinomial likelihood (Sections 7.9–7.11). We highlight advantages and disadvantages of each approach. Moreover, we will repeatedly use the generalized linear (mixed) model (GLM and GLMM) formulations to describe structure in the parameters. This allows the modeling of individual and temporal effects, both of which can either be categorical or continuous, as well as fixed or random. We will also see how correlations among parameters can be modeled through correlated random effects, using a multivariate normal distribution. In most examples, we focus on the modeling of survival, yet, clearly, similar modeling can be conducted for the recapture probability, and all these GLM formulations can also be used in other capture–recapture types of models starting with Chapter 6. Finally, in Section 7.10, we introduce posterior predictive model checking (see Gelman et al., 1996, 2004; Kéry, 2010). This provides a very general framework for the assessment of goodness-of-fit of a model to a data set.

7.2 THE CJS MODEL AS A STATE-SPACE MODEL

The state-space formulation of the CJS model has been introduced by Gimenez et al. (2007) and Royle (2008). Let us assume an individual marked at time t. It may survive until time $t+1$ with probability ϕ_t. Conceptually, we can imagine the individual tossing a coin to determine whether it survives (with probability ϕ_t) or dies (with probability $1-\phi_t$). Given that the individual is still alive at time $t+1$, it may again survive until $t+2$ with probability ϕ_{t+1}. This process is continued until the individual is either dead or the study ends. Clearly, once an individual is dead, its fate is no longer stochastic, and it will remain dead with probability 1. This is the description of the state process, that is, of the states (alive, dead) of an individual over time. We would like to know survival, which requires knowledge of these states of the individuals. Yet, we typically do not have complete information about the true states. A marked individual that is alive at occasion t may be recaptured (or more generally re-encountered) with probability p_t. Again, we can imagine that a coin is tossed, determining whether the individual is recaptured (with probability p_t) or not (with probability $1-p_t$). Once an individual is dead, it cannot be recaptured anymore. This is the description of the observation process, which is conditional on the state process, and thus there is a hierarchal structure in the state-space model. In Fig. 7.1, the two processes are shown graphically.

The data observed in a capture–recapture study can be summarized in a capture-history matrix ($\mathbf{y}$), which has dimension $I \times T$, where I is the total number of marked individuals, and T is the number of capture

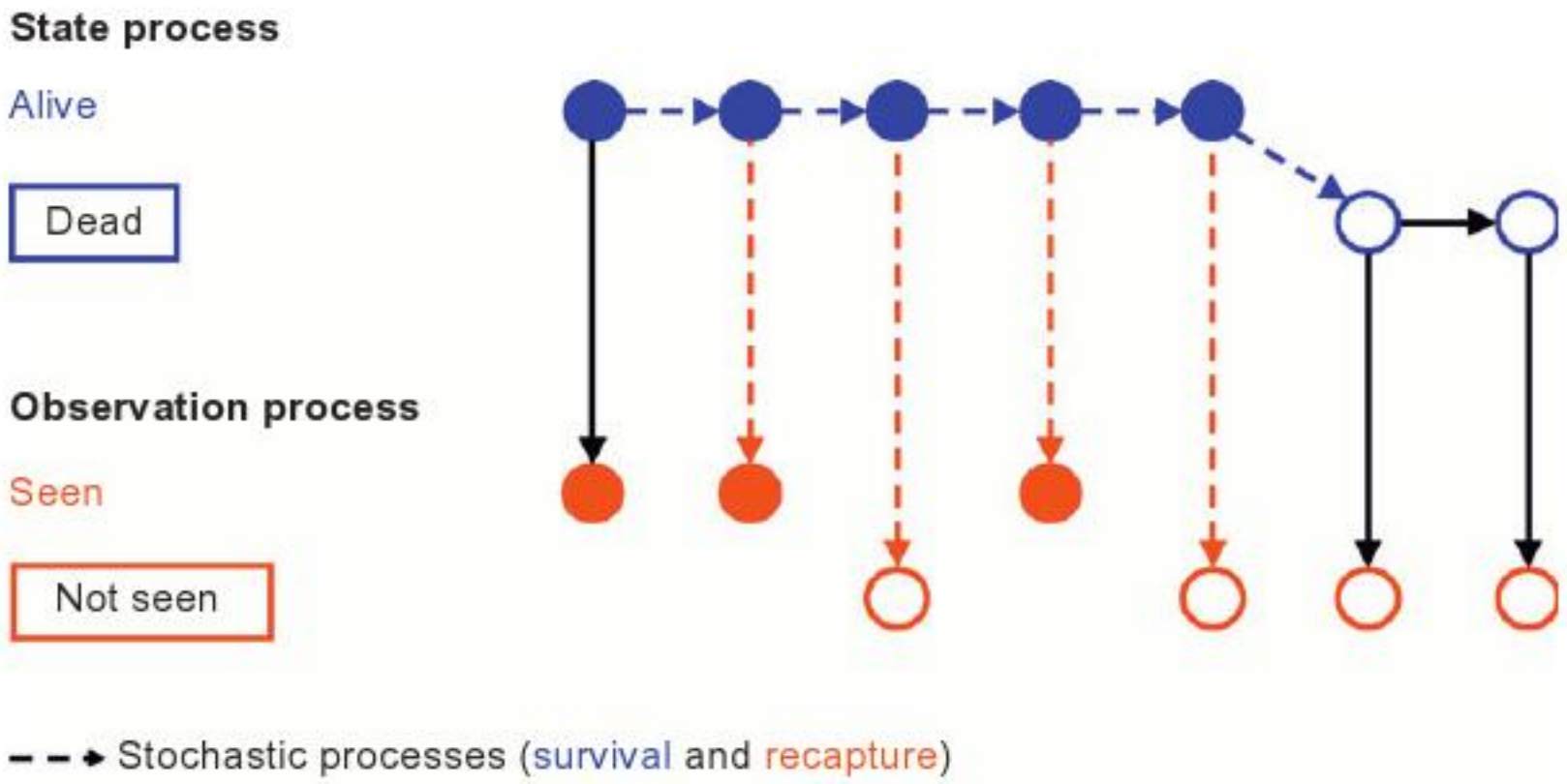

FIGURE 7.1 Example of the state and observation process of a marked individual over time for the CJS model. The sequence of true states in this individual is $z = [1, 1, 1, 1, 1, 0, 0]$, and the observed capture-history is $y = [1, 1, 0, 1, 0, 0, 0]$.

occasions. The matrix entries are either a 1 or a 0. A 1 at position i, t indicates that individual i was captured at occasion t, meaning that it was alive for sure; a 0 at position i, t shows that individual i was not captured at t, meaning that it was either dead, or alive but not caught, or not yet marked.

To estimate survival from such data, we define the latent variable $z_{i,t}$, which takes value 1 if individual i is alive at time t, and value 0 if it is dead. Thus, $z_{i,t}$ defines the true state of individual i at time t. We also define vector f_i, which denotes the occasion at which individual i is first captured (i.e., marked) because only events after first capture are modeled in the CJS model. The state of individual i at first capture ($z_{i,fi}$) is 1 with probability 1, as the individual is alive for certain. The states on subsequent occasions are modeled as Bernoulli trials. Conditional on being alive at occasion t, individual i may survive until occasion $t+1$ with probability $\phi_{i,t}$ ($t = 1, \ldots, T-1$). The following two equations define the state process:

$$z_{i,f_i} = 1$$

$$z_{i,t+1} \,|\, z_{i,t} \sim \text{Bernoulli}(z_{i,t}\phi_{i,t}).$$

The Bernoulli success parameter is composed of the product of survival and the state variable z. The inclusion of z ensures that a dead individual ($z = 0$) remains dead and has no further impact on the estimation of survival.

If individual i is alive at occasion t, it may be recaptured with probability $p_{i,t}$ ($t = 2, \ldots, T$). This can again be modeled as the realization of a

Bernoulli trial with success probability $p_{i,t}$. The following equation defines the observation process:

$$y_{i,t} \,|\, z_{i,t} \sim \text{Bernoulli}(z_{i,t} p_{i,t}).$$

The inclusion of the latent variable z in the Bernoulli trial ensures that dead individuals cannot be encountered. The state and the observation process are both defined for $t \geq f_i$. We repeat that the initial capture process is not modeled in the CJS model (see Fig. 7.1) because the initial observation at the time of capture does not contain any information about survival. In contrast, capture-histories at and before initial capture contain information about recruitment, which is a target of estimation of the Jolly–Seber models (see Chapter 10). Because initial capture is not modeled in the CJS model, we say that we condition on first capture.

The implementation of the CJS model in WinBUGS is straightforward. The most general likelihood is based on the above-mentioned three equations and contains different survival and recapture probabilities for each individual at each capture occasion. However, the parameters of this saturated model are not separately estimable (see Section 7.9 for more on this topic), and we need to introduce constraints. These constraints define the structure of the model fitted and may be imposed either along the time or the individual axis of the capture-history matrix, or along both (see Section 6.2). Thus, whatever model we fit, we do not need to change the likelihood, which describes the basic structure of the model, but just these constraints and the corresponding priors. This may not result in code that is the most efficient in terms of computing time and the easiest to read for a beginner. However, with a little practice, it will be seen to be an efficient way of fitting a wide array of models.

7.3 MODELS WITH CONSTANT PARAMETERS

We start with a very simple model, in which survival and recapture, respectively, are identical for all individuals at all occasions. Thus, we impose constraints along both the time and the individual axis of the capture-history matrix. We first simulate the data, and then analyze them. The function to simulate capture–recapture data (`simul.cjs`) is very general and works for all examples in this chapter. We choose the number of individuals released at each occasion. The function then evaluates for each released individual whether it survives, and if so, whether it is recaptured, by two Bernoulli trials governed by individual- and time-specific survival and recapture probabilities that we also provide as input (matrices `PHI` and `P`). Thus, the data-generating function works analogous to the analyzing model.

FIGURE 7.2 Pair of little owls (*Athene noctua*) (Photograph by H. Sylvain).

In the simulation, we will mimic a study on little owls (Fig. 7.2), a small owl species living in semi-open habitats such as orchards, where it likes to occupy nest boxes. Nest boxes in a study area are checked in May in six study years, and breeding adults are ringed. In each of the six study years, 50 unmarked (new) adults are caught, along with a variable number of individuals that are already marked. Survival of adult little owls is typically around 0.65 (Schaub et al., 2006), and we assume a recapture probability of 0.4. The following R code simulates a matrix with capture-histories. We do not consider individuals first captured on the last occasion, because they do not provide information about survival and recapture.

```
# Define parameter values
n.occasions <- 6                    # Number of capture occasions
marked <- rep(50, n.occasions-1)    # Annual number of newly marked
                                      individuals

phi <- rep(0.65, n.occasions-1)
p <- rep(0.4, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked))
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))

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

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

# Execute function
CH <- simul.cjs(PHI, P, marked)
```

Next, we need to create vector f, which contains the occasion at which each individual is marked.

```
# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

Finally, we write the BUGS code for a constant model. The two linear models applied are $\phi_{i,t} = \overline{\phi}$ and $p_{i,t} = \overline{p}$. These are in fact the linear predictors (see Chapter 3), but here we call them constraints because we reduce the dimensions of the $\phi_{i,t}$ and $p_{i,t}$, that is, we constrain them. We do not include covariates or random effects, so there is no need for a transformation, and the identity link is applied. The uniform priors ensure that the parameter estimates are in the interval [0, 1]. The specification of noninformative priors is easy because a uniform (U(0, 1)) or a beta distribution (beta(1,1)) can be used. Note that the time indexing in the "Likelihood" part is slightly different to that used in the formulas (see Section 7.2). It avoids the use of separate loops for the state and the observation process.

```
# Specify model in BUGS language
sink("cjs-c-c.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        phi[i,t] <- mean.phi
```

```
        p[i,t] <- mean.p
        } #t
    } #i

mean.phi ~ dunif(0, 1)          # Prior for mean survival
mean.p ~ dunif(0, 1)            # Prior for mean recapture

# Likelihood
for (i in 1:nind){
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
        mu2[i,t] <- p[i,t-1] * z[i,t]
        } #t
    } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions =
   dim(CH)[2])
```

Initial values should be given for the two structural parameters and for the latent variable z. The easiest way for the latter is just to use the observed capture-histories. We have to make sure that initial values for z are provided only after initial capture. The function below creates the required initial values based on the observed capture-histories and the vector with the occasion of first capture.

```
# Function to create a matrix of initial values for latent state z
ch.init <- function(ch, f){
    for (i in 1:dim(ch)[1]){ch[i,1:f[i]] <- NA}
    return(ch)
    }

# Initial values
inits <- function(){list(z = ch.init(CH, f), mean.phi = runif(1, 0, 1),
   mean.p = runif(1, 0, 1))}

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

# MCMC settings
ni <- 10000
nt <- 6
nb <- 5000
nc <- 3
```

```
# Call WinBUGS from R (BRT 1 min)
cjs.c.c <- bugs(bugs.data, inits, parameters, "cjs-c-c.bug", n.chains =
   nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())
```

The model does not take a long time to run and convergence is reached after just 5000 iterations. The estimates are very close to the values used for the simulations.

```
# Summarize posteriors
print(cjs.c.c, digits = 3)

           mean    sd  2.5%   25%   50%   75% 97.5%  Rhat  n.eff
mean.phi 0.679 0.043 0.601 0.649 0.677 0.707 0.767 1.003    980
mean.p   0.370 0.044 0.286 0.340 0.369 0.400 0.458 1.003   1700
```

Sometimes not all individuals recaptured are released again, for instance, when an individual dies at capture. For these individuals, we know that they have survived from initial capture until the last capture; afterward they are not in the sample anymore. This is easy to model and just requires that we define a vector h which for each individual contains the occasion after which it is not released. For individuals that stay in the sample until the end of the study, the element of vector h is just the last occasion of the study. Then, vector h must become an element of the input data and the loop in the likelihood needs to be changed from `for (t in (f[i]+1):n.occasions){ ...` to `for (t in (f[i]+1):h[i]){...`

7.3.1 Inclusion of Information about Latent State Variable

Written as state-space models, CJS models can take a long time to run because there is a loop over all individuals and occasions. The latent state variable z needs to be updated (estimated) at each MCMC iteration. So far we have treated z as if we had no information about it. The only information that we included are the observed capture-histories (Y), but they are related to z only through the observation process in the state-space model. Therefore, all elements of z (i.e., for all individuals after first capture) must be estimated, even when some of them are known.

To improve computation speed and convergence, we can add what we know about the latent state z, namely, whenever we observe a marked individual we know its latent state is $z = 1$. In addition, we know that $z = 1$ for all occasions between the first and the last observation of an individual, even if it was not seen at all occasions. To include this information in the model (i.e. to prevent estimation of what is not an unknown quantity), we create a matrix that has a value of 1 at all occasions where we know individuals were alive, and NAs elsewhere. The CJS model is conditional on first capture, so the latent state is only defined after first

capture, and thus at all first captures, we need NAs as well. The following function creates the required matrix.

```
# Function to create a matrix with information about known latent state z
known.state.cjs <- function(ch){
    state <- ch
    for (i in 1:dim(ch)[1]){
        n1 <- min(which(ch[i,]==1))
        n2 <- max(which(ch[i,]==1))
        state[i,n1:n2] <- 1
        state[i,n1] <- NA
        }
    state[state==0] <- NA
    return(state)
    }
```


This information about z is then given as data as well.

```
# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
   z = known.state.cjs(CH))
```

The initial values for z now also require some changes: we should not give initial values for those elements of z whose value is specified in the data; they get an NA.

```
# Function to create a matrix of initial values for latent state z
cjs.init.z <- function(ch,f){
    for (i in 1:dim(ch)[1]){
        if (sum(ch[i,])==1) next
        n2 <- max(which(ch[i,]==1))
        ch[i,f[i]:n2] <- NA
        }
    for (i in 1:dim(ch)[1]){
    ch[i,1:f[i]] <- NA
    }
    return(ch)
    }
```


Now, we give initial values for all the quantities to be estimated and run the model:

```
# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
   mean.p = runif(1, 0, 1))}

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

# MCMC settings
ni <- 10000
nt <- 6
```

```
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
cjs.c.c <- bugs(bugs.data, inits, parameters, "cjs-c-c.bug", n.chains =
   nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())

# Summarize posteriors
print(cjs.c.c, digits = 3)

          mean    sd  2.5%   25%   50%   75% 97.5%  Rhat  n.eff
mean.phi 0.675 0.040 0.599 0.648 0.675 0.702 0.754 1.003   2500
mean.p   0.372 0.043 0.294 0.342 0.371 0.399 0.461 1.003   2500
```

The model now runs faster. The difference in run time in this simple case is slight, but time savings can be substantial with more complex models and larger data sets. Therefore, we recommend providing all available information about the latent state in the data. In the following sections of this chapter, we will always, and in most other chapters often, do this. Note that the inclusion of the information about the latent state z has nothing to do with the use of an informative prior, we simply avoid estimation of known quantities to speed up computation.

7.4 MODELS WITH TIME-VARIATION

So far we have fitted the simplest possible model in the CJS family. It assumes that survival and recapture probabilities remain constant over time and are identical for all individuals. In practice, we typically want to relax these strict assumptions. We also may have an interest in fitting models that combine time and individual effects and modeling these effects as additive or interactive. We next consider models with temporal variation, that is, we model the column dimension of the capture-history matrix and constrain the row dimension to be constant (all individuals are treated as identical).

The variation of survival probability from one year to another often has a strong impact on the dynamics of a population. If survival varies much from year to year (i.e., temporal variability is large), population size changes more than when survival probability changes only little over time, all other demographic processes being equal. Thus, there is an interest in measuring temporal variation. Moreover, the annual fluctuations of survival or recapture may be caused by environmental factors that we may have an interest in identifying.

The models to study temporal effects of survival or recapture assume either fixed or random temporal effects, as well as the relationship between focal parameters and temporally varying covariates (e.g., weather).

The fixed-effect time model assumes the parameters to be different at each occasion and independent of each other. This approach is used if there is interest in estimates from particular occasions. By contrast, the model that considers time to be a random effect assumes that time effects are drawn from a statistical distribution, whose parameters we aim to estimate; typically, we will use a normal distribution and estimate a mean and a variance. Therefore, annual estimates are no longer independent from one another. Interest is then not so much in the individual annual effects, but more in an estimation of the mean and the variance of the annual estimates. Fixed- and random-effects models are easily fit within the framework that we have set up (see Chapter 4). The likelihood part of the BUGS code does not need any change at all: all required modifications take place in the "Priors and constraints" section of the BUGS code. In the examples that follow, we will usually model effects on survival only, but of course similar models can be adopted for recapture, too, and any combinations are possible, for example, survival with random time effects and recapture with fixed time effects.

7.4.1 Fixed Time Effects

We now assume that survival and recapture vary independently over time, that is, we regard time as a fixed-effects factor. To implement this model, we impose the following constraints: $\phi_{i,t} = \alpha_t$ and $p_{i,t} = \beta_t$, where α_t and β_t are the time-specific survival and recapture probabilities, respectively. Here is the part of the BUGS model specification that needs to be changed.

```
# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        phi[i,t] <- alpha[t]
        p[i,t] <- beta[t]
        } #t
    } #i
for (t in 1:n.occasions-1)){
    alpha[t] ~ dunif(0, 1)          # Priors for time-spec. survival
    beta[t] ~ dunif(0, 1)           # Priors for time-spec. recapture
    }
```

7.4.2 Random Time Effects

The model just shown treats time as a fixed-effects factor; for every occasion, an independent effect is estimated. To assess the temporal variability, we cannot simply take these fixed-effects estimates and calculate their variance. By doing so, we would ignore the fact that these values

are estimates that have an unknown associated error. Thus, we would assume that there is no sampling variance, and this can hardly ever be true (see, e.g., Gould and Nichols, 1998). However, when treating time as a random-effects factor, we can separate sampling (i.e., variance within years) from process variance (i.e., variance between years), exactly as we did in the state-space models in Chapter 5. We model survival or recapture probabilities on the logit scale as a realization of a random process described by a normal distribution with mean μ and variance σ^2. The logit link function ensures that the estimated probabilities remain within the interval between 0 and 1:

$$\text{logit}(\phi_{i,t}) = \mu + \varepsilon_t$$
$$\varepsilon_t \sim \text{Normal}(0, \sigma^2).$$

ε_t is the deviation from the overall mean survival probability; thus it is a "temporal residual". The temporal variance (σ^2) is on the logit scale; thus, it is the temporal variance of the logit survival. Sometimes, one needs an estimate on the probability scale, for instance, when the temporal variance should be compared with the variance of another demographic rate to decide which parameter is more variable over time. A back-transformation is possible by applying the delta method (Powell, 2007). We use

$$\sigma^2_\theta \cong \sigma^2 \theta^2 (1-\theta)^2,$$

where $\theta = \frac{\exp(\mu)}{1+\exp(\mu)}$ and σ^2_θ is the variance on the back-transformed scale. It is easy to estimate this quantity directly in BUGS.

To illustrate the approach, we simulate data and analyze them. In the little owl example, we assume a mean survival probability of females of 0.65 and temporal variance of 1 on the logit scale. Reasonable estimates of the temporal variance require a large number of years (>10; Burnham and White, 2002). Here, we simulate data over 20 years.


```
# Define parameter values
n.occasions <- 20                    # Number of capture occasions
marked <- rep(30, n.occasions-1)     # Annual number of newly marked
                                       individuals
mean.phi <- 0.65
var.phi <- 1                         # Temporal variance of survival
p <- rep(0.4, n.occasions-1)

# Determine annual survival probabilities
logit.phi <- rnorm(n.occasions-1, qlogis(mean.phi), var.phi^0.5)
phi <- plogis(logit.phi)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked), byrow = TRUE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```
# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

In the BUGS model description, we only alter parts in the "Priors and constraints" sections; no change is required in the likelihood part. In particular, we have to implement the random-effects formulation (formula above). The prior choices for μ and for σ^2 need some thought. Because μ is the mean survival on the logit scale, a noninformative prior on the logit scale would be a normal distribution with a wide variance. Yet, this prior will not be noninformative on the probability scale. In the code below, we provide two options: first, a normal distribution with a wide variance for μ, and second, a uniform distribution for $\text{logit}^{-1}(\mu)$, which is noninformative on the probability scale but informative on the logit scale. A prior is also needed for σ^2. Following Gelman (2006), we use a uniform distribution for the standard deviation because this induces little information. We will fit the same model under which we generated the data, that is, model $\phi_{\underline{t}}$, $p_.$, where by the underlined index for time, we denote random time effects.

```
# Specify model in BUGS language
sink("cjs-temp-raneff.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        logit(phi[i,t]) <- mu + epsilon[t]
        p[i,t] <- mean.p
        } #t
    } #i
for (t in 1:(n.occasions-1)){
    epsilon[t] ~ dnorm(0, tau)
    }

#mu ~ dnorm(0, 0.001)                    # Prior for logit of mean survival
#mean.phi <- 1 / (1+exp(-mu))            # Logit transformation
mean.phi ~ dunif(0, 1)                   # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi))       # Logit transformation
sigma ~ dunif(0, 10)                     # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)                  # Temporal variance
mean.p ~ dunif(0, 1)                     # Prior for mean recapture

# Likelihood
for (i in 1:nind){
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
```

```
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
        mu2[i,t] <- p[i,t-1] * z[i,t]
        } #t
    } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
   z = known.state.cjs(CH))

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
   sigma = runif(1, 0, 10), mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p", "sigma2")

# MCMC settings
ni <- 10000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 17 min)
cjs.ran <- bugs(bugs.data, inits, parameters, "cjs-temp-raneff.bug",
   n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())
```

The chains evolve slowly and convergence is not achieved swiftly. This can be improved if a smaller range is chosen for the prior for the standard deviation of the temporal variance. Here, we used a uniform prior in the interval between 0 and 10, thus the variance could take values between 0 and 100. Had we chosen a higher upper bound, the estimate would probability not change (recall we simulated the data with a variance of 1), but the chains would converge even more slowly. Computation for this model is more efficient for the multinomial formulation of the model (see Section 7.10).

```
# Summarize posteriors
print(cjs.ran, digits = 3)

          mean    sd  2.5%   25%   50%   75% 97.5%  Rhat  n.eff
mean.phi 0.634 0.073 0.488 0.590 0.634 0.678 0.787 1.017    120
mean.p   0.394 0.024 0.350 0.378 0.394 0.411 0.441 1.001   2000
sigma2   1.700 1.152 0.548 1.006 1.402 2.005 4.687 1.006    440

# Produce histogram
hist(cjs.ran$sims.list$sigma2, col = "gray", nclass = 35, las = 1,
   xlab = expression(sigma^2), main = "")
abline(v = var.phi, col = "red", lwd = 2)
```

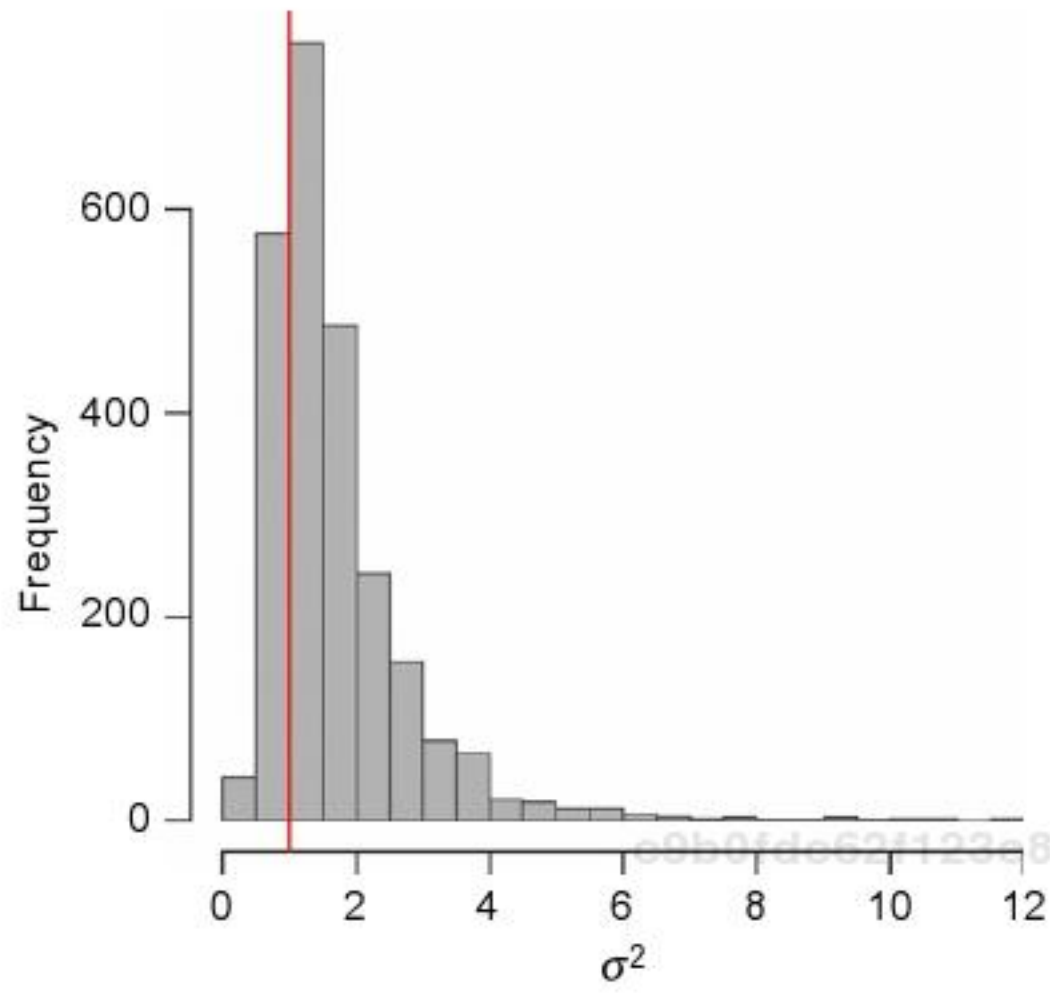

FIGURE 7.3 Posterior distribution of the temporal variance in apparent survival (red: value used for data generation).

The histogram of the posterior samples of the temporal variance for one simulated data set is shown in Fig. 7.3. The point estimate may seem biased; however, recall that this is just a single simulation, and we would need many simulations to check for any bias (see exercise 4 in Section 7.13).

7.4.3 Temporal Covariates

Often we are not only interested in getting a point estimate of survival or of its temporal variability, but also in identifying factors affecting survival. One way to do this is to see whether the observed temporal pattern in survival matches the temporal variation of an environmental factor (e.g., winter severity). From a nonzero correlation we would then infer an effect of that factor on survival. However, regardless of how the model is specified, such evidence is of correlative nature, thus causation cannot be inferred. A properly designed experiment is needed to infer causation, which is not easy in population studies (but see Schwarz, 2002).

Traditionally, so-called ultrastructural modeling has been used to model survival as a function of a covariate (x) (Lebreton et al., 1992; Link, 1999):

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_t.$$

This model assumes that the entire temporal variability of survival could be explained by the covariate x; it is analogous to a linear regression model without residuals. This seems quite unrealistic, but in earlier times, this was the only way that the relationship between survival and a covariate could be modeled. A more realistic approach is to assume that only part of the temporal variability of survival is explained by the covariate, another part being unexplained random variation. Thus, we specify this model:

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_t + \varepsilon_t$$

$$\varepsilon_t \sim \text{Normal}(0, \sigma^2).$$

The residual variance (σ^2) is the unexplained temporal variance. This allows us to estimate the amount of the total temporal variance which is explained by covariate x. We need to fit a model without the covariate to get an estimate of the total temporal variance (σ^2_{total}). The proportion of the variance explained by covariate x is then $(\sigma^2_{\text{total}} - \sigma^2)/\sigma^2_{\text{total}}$ (Grosbois et al., 2008).

To illustrate the model with the little owl example, we assume a mean survival of 0.65 and a negative effect of winter severity with logistic-linear slope of −0.3. The winter severity index is standardized (mean = 0, variance = 1) and the residual temporal variance not explained by winter severity has variance of 0.2.

```
# Define parameter values
n.occasions <- 20                          # Number of capture occasions
marked <- rep(15, n.occasions-1)           # Annual number of newly marked
                                             individuals

mean.phi <- 0.65
p <- rep(0.4, n.occasions-1)
beta <- -0.3                               # Slope of survival-winter
                                             relationship
r.var <- 0.2                               # Residual temporal variance

# Draw annual survival probabilities
winter <- rnorm(n.occasions-1, 0, 1^0.5)
logit.phi <- qlogis(mean.phi) + beta*winter + rnorm(n.occasions-1, 0,
   r.var^0.5)
phi <- plogis(logit.phi)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked),
   byrow = TRUE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```
# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Specify model in BUGS language
sink("cjs-cov-raneff.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
   for (t in f[i]:(n.occasions-1)){
      logit(phi[i,t]) <- mu + beta*x[t] + epsilon[t]
      p[i,t] <- mean.p
      } #t
   } #i
for (t in 1:(n.occasions-1)){
   epsilon[t] ~ dnorm(0, tau)
   phi.est[t] <- 1 / (1+exp(-mu-beta*x[t]-epsilon[t])) # Yearly
                                                          survival
   }
mu ~ dnorm(0, 0.001)                  # Prior for logit of mean survival
mean.phi <- 1 / (1+exp(-mu))          # Logit transformation
beta ~ dnorm(0, 0.001)I(-10, 10)      # Prior for slope parameter
sigma ~ dunif(0, 10)                  # Prior on standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)               # Residual temporal variance
mean.p ~ dunif(0, 1)                  # Prior for mean recapture

# Likelihood
for (i in 1:nind){
   # Define latent state at first capture
   z[i,f[i]] <- 1
   for (t in (f[i]+1):n.occasions){
      # State process
      z[i,t] ~ dbern(mu1[i,t])
      mu1[i,t] <- phi[i,t-1] * z[i,t-1]
      # Observation process
      y[i,t] ~ dbern(mu2[i,t])
      mu2[i,t] <- p[i,t-1] * z[i,t]
      } #t
   } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
   z = known.state.cjs(CH), x = winter)
```

```
# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mu = rnorm(1), sigma =
   runif(1, 0, 5), beta = runif(1, -5, 5), mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p", "phi.est", "sigma2", "beta")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 12 min)
cjs.cov <- bugs(bugs.data, inits, parameters, "cjs-cov-raneff.bug",
   n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())
```


In general, models with random effects are more difficult to fit, and we need to run long Markov chains to achieve satisfactory convergence for all parameters. The posterior distributions of the slope and the residual environmental variation (temporal variation not explained by the covariate) are shown in Fig. 7.4.

```
# Summarize posteriors
print(cjs.cov, digits = 3)
```

```
                mean     sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
mean.phi       0.707  0.050  0.611  0.676  0.705  0.736  0.805 1.012  1400
mean.p         0.403  0.032  0.343  0.381  0.403  0.424  0.467 1.002  1800
phi.est[1]     0.686  0.121  0.423  0.610  0.696  0.769  0.902 1.003  1200
[ ... ]
```

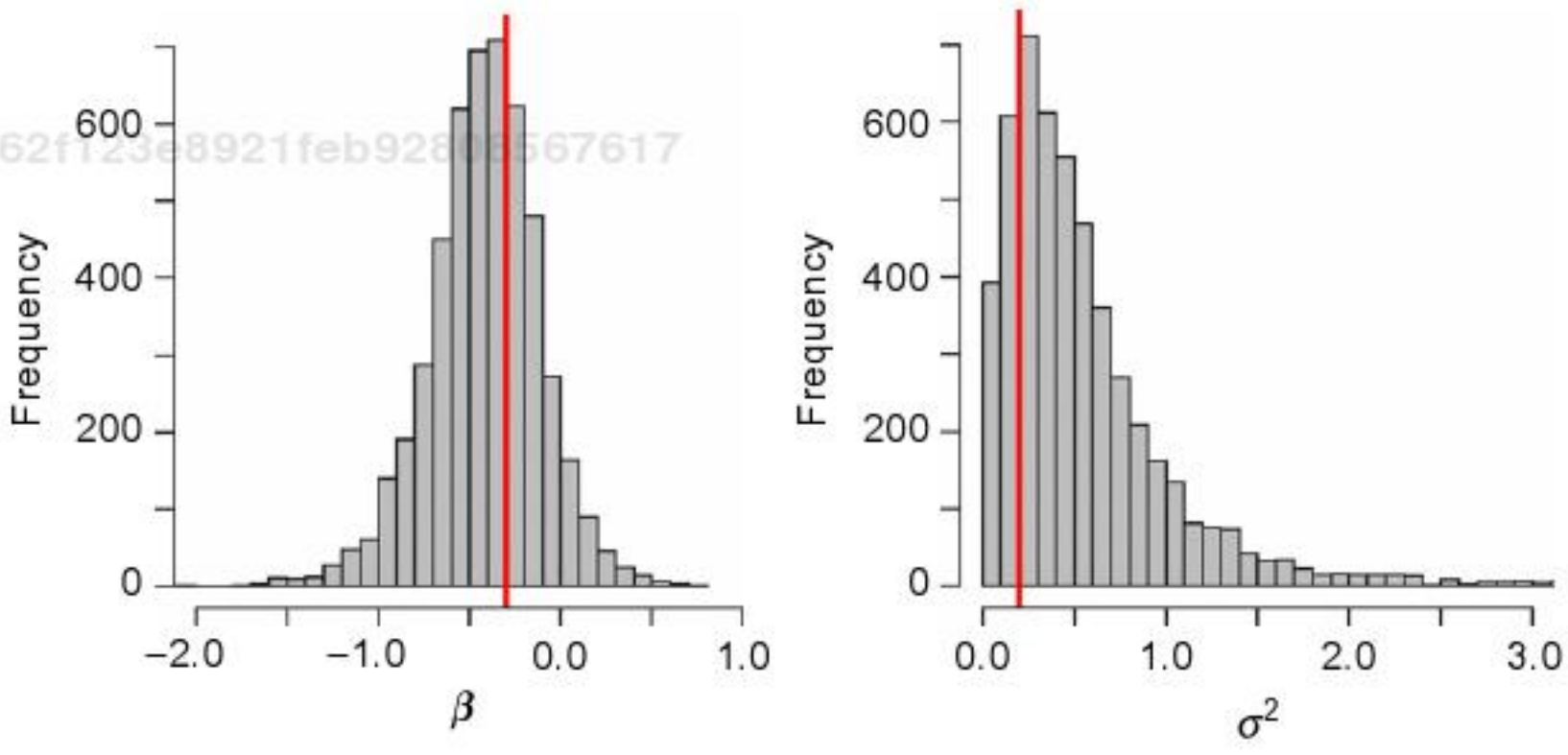

FIGURE 7.4 Posterior distributions of the covariate effect (slope parameter β) and of environmental variability (the residual temporal variance σ^2). Red lines indicate the values used for simulating the data.

```
phi.est[19]   0.681  0.120   0.427   0.603   0.690   0.764  0.902 1.003    950
sigma2        0.566  0.541   0.047   0.234   0.426   0.714  2.012 1.008    390
beta         -0.422  0.308  -1.080  -0.600  -0.410  -0.226  0.160 1.006   1400
```

```
# Produce graph
par(mfrow = c(1, 2), las = 1)
hist(cjs.cov$sims.list$beta, nclass = 25, col = "gray", main = "",
   xlab = expression(beta), ylab = "Frequency")
abline(v = -0.3, col = "red", lwd = 2)
hist(cjs.cov$sims.list$sigma2, nclass = 50, col = "gray", main = "",
   xlab = expression(sigma^2), ylab = "Frequency", xlim=c(0, 3))
abline(v = 0.2, col = "red", lwd = 2)
```

7.5 MODELS WITH INDIVIDUAL VARIATION

So far we have modeled temporal effects, assuming identical survival and recapture for all individuals. Now, we relax this assumption and model individual heterogeneity. Thus, we model effects along the row axis of the capture-history matrix. As with models for time effects, we can model individual effects in different ways—individual effects can be categorical or continuous, fixed or random, or latent or explained by measured covariates (compare with model M_h in Section 6.2). Moreover, individual effects can be constant over time, or they may change over time. Here, we only consider the simpler case, where they are constant over time.

7.5.1 Fixed Group Effects

We may specify fixed effects when we are interested in the estimates of particular groups (e.g., sex) and if the number of groups is low, as it is difficult to estimate the between-group variance with a small number of groups. We define g_i as a categorical variable with G levels (i.e., number of groups) indicating the group membership. The model for survival with a fixed group effect is

$$\phi_{i,t} = \beta_{g(i)},$$

where index $g(i)$ denotes the group g to which individual i belongs and β_g $(g = 1 \ldots G)$ are the estimated fixed group effects. Because there are no other effects in the model, we can model directly on the probability scale, and the prior distribution ensures that the parameter estimates are between 0 and 1.

We illustrate the model to estimate sex-specific survival and recapture probabilities with simulated data of little owls. We first simulate two separate capture–recapture data sets; one for males and another for females.

Then we create the grouping variable g (named "group") and merge the two capture–recapture data sets.

```
# Define parameter values
n.occasions <- 12                          # Number of capture occasions
marked <- rep(30, n.occasions-1)           # Annual number of newly marked
                                             individuals
phi.f <- rep(0.65, n.occasions-1)          # Survival of females
p.f <- rep(0.6, n.occasions-1)             # Recapture prob. of females
phi.m <- rep(0.8, n.occasions-1)           # Survival of males
p.m <- rep(0.3, n.occasions-1)             # Recapture prob. of males

# Define matrices with survival and recapture probabilities
PHI.F <- matrix(phi.f, ncol = n.occasions-1, nrow = sum(marked))
P.F <- matrix(p.f, ncol = n.occasions-1, nrow = sum(marked))
PHI.M <- matrix(phi.m, ncol = n.occasions-1, nrow = sum(marked))
P.M <- matrix(p.m, ncol = n.occasions-1, nrow = sum(marked))

# Simulate capture-histories
CH.F <- simul.cjs(PHI.F, P.F, marked)
CH.M <- simul.cjs(PHI.M, P.M, marked)

# Merge capture-histories by row
CH <- rbind(CH.F, CH.M)

# Create group variable
group <- c(rep(1, dim(CH.F)[1]), rep(2, dim(CH.M)[1]))

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

Finally, we write the model in BUGS language and fit it to the data.

```
# Specify model in BUGS language
sink("cjs-group.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
   for (t in f[i]:(n.occasions-1)){
      phi[i,t] <- phi.g[group[i]]
      p[i,t] <- p.g[group[i]]
      } #t
   } #i
for (u in 1:g){
   phi.g[u] ~ dunif(0, 1)              # Priors for group-specific
                                         survival
   p.g[u] ~ dunif(0, 1)                # Priors for group-specific
                                         recapture
   }

# Likelihood
for (i in 1:nind){
```

```
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
        mu2[i,t] <- p[i,t-1] * z[i,t]
        } #t
    } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
   z = known.state.cjs(CH), g = length(unique(group)), group = group)

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), phi.g = runif(length
   (unique(group)), 0, 1), p.g = runif(length(unique(group)), 0, 1))}

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

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 2 min)
cjs.group <- bugs(bugs.data, inits, parameters, "cjs-group.bug",
   n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())
```

The parameter estimates are close to the values used to generate the data.

```
# Summarize posteriors
print(cjs.group, digits = 3)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
phi.g[1]	0.656	0.020	0.617	0.642	0.656	0.669	0.694	1.001	3000
phi.g[2]	0.796	0.018	0.760	0.784	0.796	0.808	0.831	1.002	1900
p.g[1]	0.599	0.029	0.541	0.578	0.599	0.619	0.654	1.002	1700
p.g[2]	0.325	0.021	0.285	0.311	0.324	0.339	0.368	1.002	1900

7.5.2 Random Group Effects

We may specify random group effects when we are interested in an overall mean and the variability between groups. A typical example of random group effects is provided by local populations, where we are

interested in estimating spatial variation of survival (Grosbois et al., 2009). Survival is then modeled as

$$\text{logit}(\phi_{i,t}) = \beta_{g(i)}$$
$$\beta_g \sim \text{Normal}(\bar{\beta}, \sigma^2),$$

where σ^2 is the variance of logit survival between groups, β_g are the random group effects, and $\bar{\beta}$ is the overall mean. Note that we now use the logit link function to ensure that the realized group-specific survival probabilities ($\text{logit}^{-1}(\beta_g)$) are bound in the interval [0, 1].

Because most BUGS code is identical to that in Section 7.5.1, we just show the part which needs modification:

```
# Priors and constraints
for (i in 1:nind){
   for (t in f[i]:(n.occasions-1)){
      logit(phi[i,t]) <- beta[group[i]]
      p[i,t] <- mean.p
      } #t
   } #i
for (u in 1:g){
   beta[u] ~ dnorm(mean.beta, tau)
   phi.g[u] <- 1 / (1+exp(-beta[u])) # Back-transformed
                                       group-specific survival
   }
mean.beta ~ dnorm(0, 0.001)          # Prior for logit of mean survival
mean.phi <- 1 / (1+exp(-mean.beta)) # Back-transformed mean survival
sigma ~ dunif(0, 10)                 # Prior for sd of logit of survival
                                       variability
tau <- pow(sigma, -2)
mean.p ~ dunif(0, 1)                 # Prior for mean recapture
```


7.5.3 Individual Random Effects

As an extreme case of a random group effect, we could also consider each individual as belonging to its own group. This model would not be identifiable when groups are treated as fixed affects, but it is when we treat groups as random effects. Conceptually, we imagine that there is an average survival, around which there is individual-specific noise. To specify individual random effects, we write

$$\text{logit}(\phi_{i,t}) = \mu + \varepsilon_i$$
$$\varepsilon_i \sim \text{Normal}(0, \sigma^2),$$

where σ^2 is the variance of logit survival among individuals, and μ is the overall mean logit survival. The interest of an analysis with individual random effects may be in estimating the mean, the variance, or even the realized survival "residuals" of each individual (sometimes called

"frailty"; Cam et al., 2002). Such a model also provides the base for modeling survival as a function of an individual covariate x_i (e.g., size of an individual). This model can be written as

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_i + \varepsilon_i$$
$$\varepsilon_i \sim \text{Normal}(0, \sigma^2),$$

where β is the slope of covariate x on logit survival.

Models with random individual variation in survival are particularly important for the study of senescence (Cam et al., 2002). If individual variation is not included, senescence could easily be overlooked because a decline with age may be offset by increasing proportions of high-quality individuals in the population (Service, 2000; van de Pol and Verhulst, 2006). Sometimes, such a model may also be adopted for recapture probabilities because they are likely to differ among individuals in a similar manner.

Capture–recapture data are often subject to overdispersion, which may be due to a lack of independence among individuals (Lebreton et al., 1992). Overdispersion can be detected with a goodness-of-fit test (Lebreton et al., 1992; Choquet et al., 2001). If overdispersion is not corrected for, parameter estimators tend to be unbiased, but their variances (e.g., standard errors) will be too small (Anderson et al., 1994). The frequentist solution is to calculate a variance inflation factor from the goodness-of-fit test that is called c-hat in the capture–recapture literature, and to compute the true variance of the estimates as the product of the apparent variance and c-hat. An analogous Bayesian solution is to use a model with individual random effects (see also chapter 14 in Kéry 2010 and Section 4.2). The advantage of the Bayesian solution is the flexibility in specifying lack of independence in either survival only, recapture only, or in both parameters.

We now simulate little owl data and analyze them. We assume mean survival of 0.65 and individual variability with a variance of 0.5.


```
# Define parameter values
n.occasions <- 20                    # Number of capture occasions
marked <- rep(30, n.occasions-1)     # Annual number of newly marked
                                       individuals
mean.phi <- 0.65
p <- rep(0.4, n.occasions-1)
v.ind <- 0.5

# Draw annual survival probabilities
logit.phi <- rnorm(sum(marked), qlogis(mean.phi), v.ind^0.5)
phi <- plogis(logit.phi)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked),
   byrow = FALSE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```
# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Specify model in BUGS language
sink("cjs-ind-raneff.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        logit(phi[i,t]) <- mu + epsilon[i]
        p[i,t] <- mean.p
        } #t
    } #i
for (i in 1:nind){
    epsilon[i] ~ dnorm(0, tau)
    }
mean.phi ~ dunif(0, 1)                  # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi))      # Logit transformation
sigma ~ dunif(0, 5)                     # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)
mean.p ~ dunif(0, 1)                    # Prior for mean recapture

# Likelihood
for (i in 1:nind){
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
        mu2[i,t] <- p[i,t-1] * z[i,t]
        } #t
    } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
   z = known.state.cjs(CH))

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
   mean.p = runif(1, 0, 1), sigma = runif(1, 0, 2))}
```

```
# Parameters monitored
parameters <- c("mean.phi", "mean.p", "sigma2")

# MCMC settings
ni <- 50000
nt <- 6
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT 73 min)
cjs.ind <- bugs(bugs.data, inits, parameters, "cjs-ind-raneff.bug",
   n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
   bugs.directory = bugs.dir, working.directory = getwd())
```

We need relatively long runs to reach satisfactory convergence. We note that the random-effects distribution could also be truncated like `epsilon[i] ~ dnorm(0, tau)I(-15, 15)` to improve mixing of the chains (see Appendix 1, tipp 16). The posterior distributions of the two parameters, mean survival and variability among individuals, show good agreement with the simulated parameters (Fig. 7.5).

```
# Summarize posteriors
print(cjs.ind, digits = 3)

          mean    sd  2.5%   25%   50%   75% 97.5%  Rhat  n.eff
mean.phi 0.640 0.026 0.587 0.623 0.641 0.658 0.688 1.001  15000
mean.p   0.410 0.021 0.368 0.396 0.410 0.424 0.451 1.001  13000
sigma2   0.586 0.244 0.176 0.410 0.560 0.739 1.132 1.012   1800

# Produce graph
par(mfrow = c(1, 2), las = 1)
hist(cjs.ind$sims.list$mean.phi, nclass = 25, col = "gray", main = "",
   xlab = expression(bar(phi)), ylab = "Frequency")
abline(v = mean.phi, col = "red", lwd = 2)
```

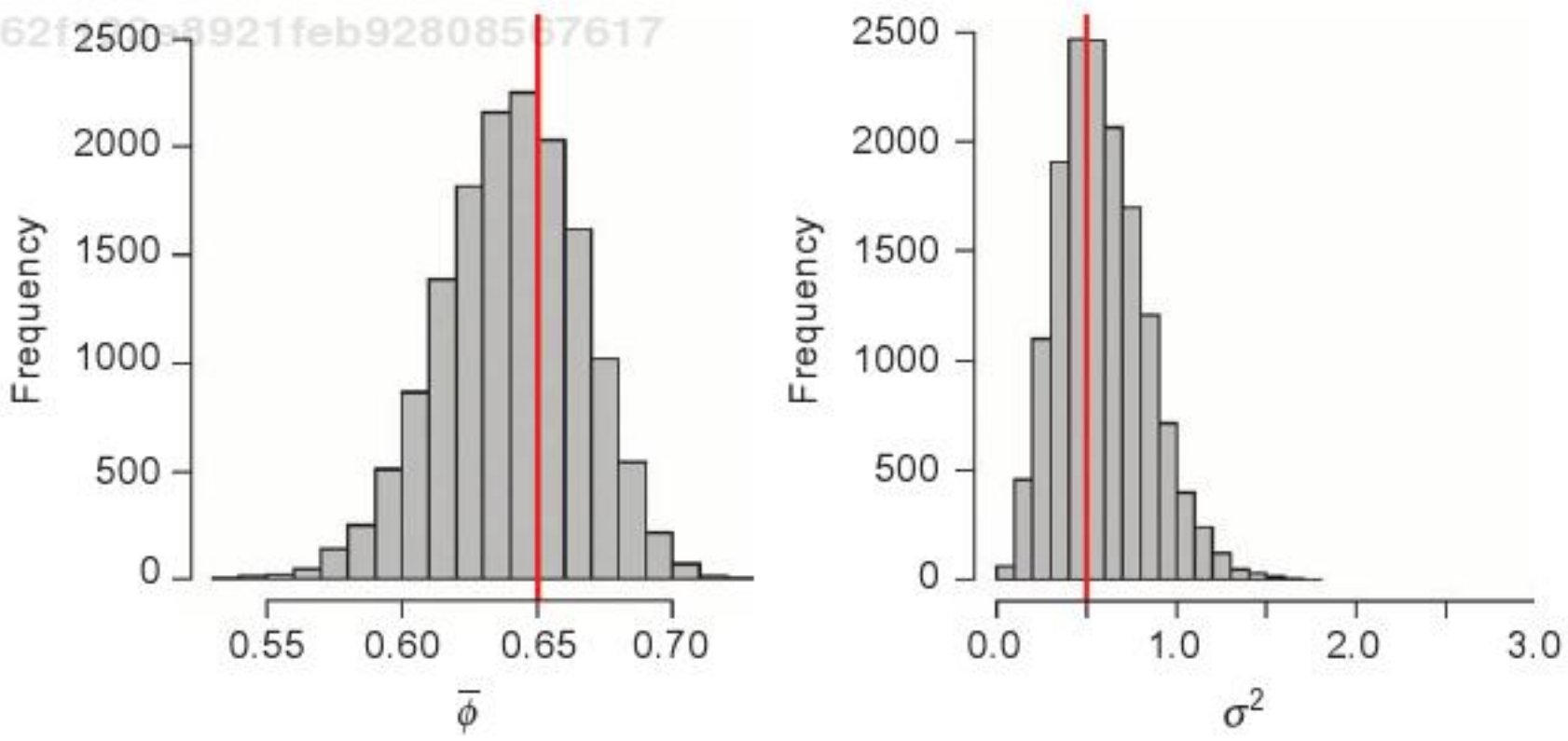

FIGURE 7.5 Posterior distributions of mean survival and of the individual variance in survival. Red lines indicate the values used for data simulation.

```
hist(cjs.ind$sims.list$sigma2, nclass = 15, col = "gray", main = "",
   xlab = expression(sigma^2), ylab = "Frequency", xlim = c(0, 3))
abline(v = v.ind, col = "red", lwd = 2)
```

If we wanted to estimate survival as a function of an individual covariate x, then we just have to adapt a small part in the code:

```
# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        logit(phi[i,t]) <- mu + beta*x[i] + epsilon[i]
        p[i,t] <- mean.p
        } #t
    } #i
for (i in 1:nind){
    epsilon[i] ~ dnorm(0, tau)
    }
mean.phi ~ dunif(0, 1)                  # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi))      # Logit transformation
beta ~ dnorm(0, 0.001)                  # Prior for covariate slope
sigma ~ dunif(0, 5)                     # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)
mean.p ~ dunif(0, 1)                    # Prior for mean recapture
```

Of course, we also have to give initial values for the new stochastic node beta, to include the covariate x in `bugs.data`, and to monitor beta.

Individual covariates may also change over time, such as, for example, body mass. The difficulty is that the covariate is unknown at occasions when the individual was not captured. Estimating the effects of individual time-varying covariates on survival is a challenge and different approaches have been proposed (Bonner and Schwarz, 2006; Catchpole et al., 2008; King et al., 2010).

7.6 MODELS WITH TIME AND GROUP EFFECTS

7.6.1 Fixed Group and Time Effects

Clearly we can combine the two concepts introduced in Sections 7.4 and 7.5 and model structure both along the time and along the individual axis of the capture-history matrix. The changes needed in the model code are merely an explicit GLM formulation of effects. This offers great flexibility as we can consider interacting or additive time and group effects, and we can treat either or both as random. The different combinations are straightforward and easy to implement, so we now focus in detail on one particular model that is often used, an additive model with fixed time and group effects.