CHAPTER

4

Introduction to Random Effects: Conventional Poisson GLMM for Count Data

OUTLINE

4.1 INTRODUCTION

In Chapter 3, we have seen that the generalized linear model (GLM) is an extremely flexible and useful model and that much of applied statistics in ecology is based on GLMs. Now, we introduce perhaps the most

Bayesian Population Analysis using WinBUGS
DOI: 10.1016/B978-0-12-387020-9.00004-3

important extension of the GLM: random effects. A GLM with fixed and random effects is also called a generalized linear mixed model (GLMM; Bolker, 2008). In this section, we explain what random effects are and describe the reasons for wanting to describe, and the consequences of declaring, a set of effects as random. After that, we look at a simple example of a Poisson GLMM, where random year effects are used to account for overdispersion in a time series of counts (Section 4.2). The resulting Poisson–log-normal model may not be the most familiar application of random effects, but it builds directly on the peregrine GLM example in Chapter 3. Furthermore, it illustrates the important topic of accounting for overdispersion in nonnormal GLMs for count data (Lee and Nelder, 2000; Millar, 2009). In Section 4.3, we fit more complex GLMMs with multiple sets of random effects. This is the kind of random effects that you may be more familiar with: latent, that is, unobserved, effects that are shared by all members of a group and that induce a correlated response among members of the same group. We call all models in this chapter "conventional" mixed models because they are not based on any underlying population model such as the mixed models in Chapter 5.

4.1.1 An Example

To illustrate, we revisit the ANCOVA example from Section 3.2.2, simply with changed variable names. We assume that we studied the mass–length relationship in asp vipers (Fig. 1.4) and had examined nine individuals in three populations. Here are the data and code for a plot.

```
# Define and plot data
mass <- c(25, 14, 68, 79, 64, 139, 49, 119, 111)
pop <- factor(c(1, 1, 1, 2, 2, 2, 3, 3, 3))
length <- c(1, 14, 22, 2, 9, 20, 2, 13, 22)
plot(length, mass, col = c(rep("red", 3), rep("blue", 3),
   rep("green", 3)), xlim = c(-1, 25), ylim = c(0, 140), cex = 1.5, lwd = 2,
   frame.plot = FALSE, las = 1, pch = 16, xlab = "Length", ylab = "Mass")
```

The plot suggests a linear relationship between mass and length with a different baseline in each population. A regression model ought to account for population differences. In one of the simplest such models, the mass of snake i depends on length in a linear way, allows populations j to have a different intercept, and has residuals ε_i come from a zero-mean normal distribution with variance σ^2.

$$\text{mass}_i = \alpha_{j(i)} + \beta * \text{length}_i + \varepsilon_i$$

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

One way to motivate random effects is to recognize that there are two assumptions that one could make about the three asp viper populations studied:

1. They are the only populations that we are interested in.
2. They merely represent a sample from a larger number of populations that we *could* have studied and we would like to generalize our conclusions to this larger statistical population of snake populations.

These assumptions about the populations can be translated into the following assumptions about the population effects α_j:

1. The three α_j are completely independent.
2. The three α_j are not independent; instead, we regard them as a sample from a larger number of effects, which *could* have appeared in our study, and we would like to learn something about that population of effects.

Traditionally, assumption 1 leads one to treat the α_j as fixed effects, while assumption 2 leads to the adoption of a random-effects model for the effects α_j. Algebraically, the *only* difference between the two models is that for the random-effects ANCOVA, we add a distributional assumption about the intercepts of the three regression lines, that is, about the population effects α_j.

$$\text{mass}_i = \alpha_{j(i)} + \beta * \text{length}_i + \varepsilon_i$$

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

$$\alpha_j \sim \text{Normal}(\mu_\alpha, \sigma_\alpha^2). \quad \text{This line makes } \alpha_j \text{ random!}$$

The third equation is the *only* difference between a model that treats the population effects α_j as fixed and one where α_j are random: without that line, the effects are fixed, and with it, they are random. Line 3 defines α_j as random draws from a normal distribution with mean μ_α and variance σ_α^2. This model is also called a random-intercepts model and is an example of a mixed model, that is, one with both random effects (α_j) and fixed effects (β). See Kéry (2010) for many further examples of fixed-effects models and their random-effects analogues; also see Section 4.3.2.

We use R to fit both models and plot the resulting regression lines (Fig. 4.1). For the random-effects model, we use the REML method, a variant of maximum likelihood that is better suited for mixed models and fit the model with the function `lmer` in the R package lme4, which we need to load first.

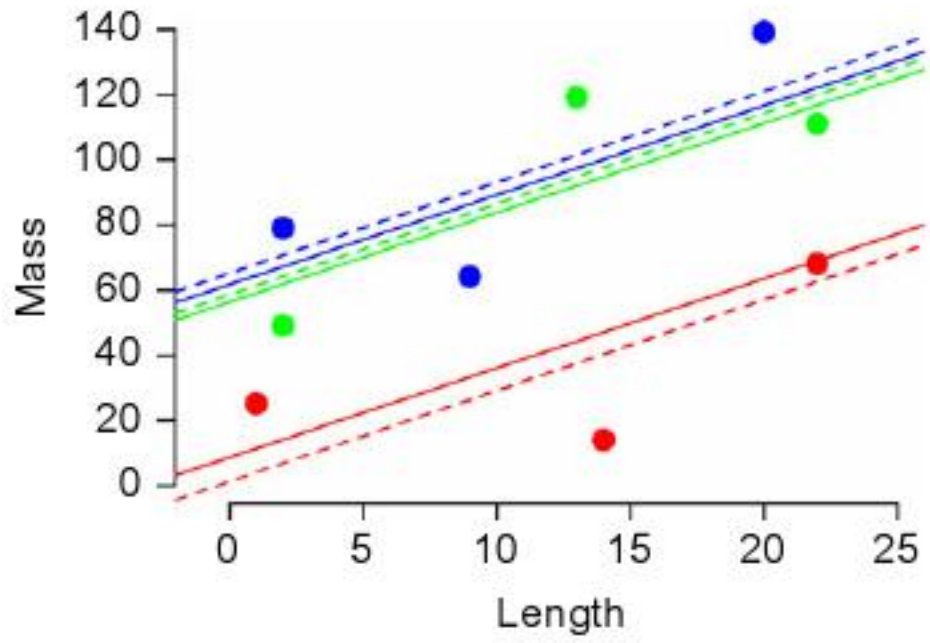

FIGURE 4.1 Relationship between mass and length of asp vipers in three populations (indicated by colors). Circles are the nine measurements. Dashed lines are the estimated regressions for each population under an ANCOVA model with fixed population effects (intercepts). Solid lines are the estimated regressions under a mixed ANCOVA model with random population effects (intercepts). Intercepts under the random-effects ANCOVA are shrunk toward the grand mean, that is, the average intercept.

```
# Fit fixed-effects model, print regression parameter estimates and
   plot regression lines
summary(lm <- lm(mass ~ pop-1 + length))
abline(lm$coef[1], lm$coef[4], col = "red", lwd = 3, lty = 2)
abline(lm$coef[2], lm$coef[4], col = "blue", lwd = 3, lty = 2)
abline(lm$coef[3], lm$coef[4], col = "green", lwd = 3, lty = 2)

# Fit mixed model, print random effects and plot regression lines
summary(lmm <- lmer(mass ~ length + (1|pop)))
ranef(lmm)
abline((lmm@fixef[1]+ranef(lmm)$pop)[1,], lmm@fixef[2], col = "red",
   lwd = 3)
abline((lmm@fixef[1]+ranef(lmm)$pop)[2,], lmm@fixef[2], col = "blue",
   lwd = 3)
abline((lmm@fixef[1]+ranef(lmm)$pop)[3,], lmm@fixef[2], col = "green",
   lwd = 3)
```


Thus, when we describe a model in algebra, the difference between a fixed- and the random-effects version of a model is really trivial. Nevertheless, random effects appear to be a fairly challenging concept for many ecologists. This may have to do with two things. First, random effects are often introduced within the context of ANOVA decompositions of sums of squares. This may simply not be the best setting within which to explain the concepts of the actual models. Second, in almost all software, there is a big divide between procedures that are used to fit the fixed- and the random-effects versions of a model. For instance, in R, there is `lm` for fixed-effects normal linear models and `lmer` for random-effects normal linear models. The two have quite a different syntax and `lmer` cannot even be used to fit fixed-effects models, and vice versa. Hence, the differences rather than the similarities between the two versions of a model are emphasized.

Fortunately, this is different with WinBUGS and the BUGS language; here, we can easily switch between fixed and random effects in a model. We have met quite a few ecologists who have only started to understand random effects after fitting random-effects models in WinBUGS. In the next sections, we try to explain in more detail random effects, the motivation for treating a set of effects as random, and why we will not always treat all effects in a model as random. When possible, we will refer back to the introductory example.

4.1.2 What Are Random Effects?

Put simply, random effects are two or more effects or parameters that "belong together" in some way. Specifically, they are estimated under the constraint of a common distribution. We may imagine that a common stochastic process has generated the effects. Therefore, replication of a study would yield a different set of realizations from that random process, that is, a different sample of effects from the distribution describing the stochastic process. In the Bayesian literature, one often reads that to assume effects as random, they must be exchangeable. Exchangeability implies similar, but not identical effects, and is a judgment, not something that can be tested. We are usually free to choose to treat a factor as fixed or random, but there are cases in which the assumption of exchangeability (treating effects as random) would not be biologically justified. For instance, if two of our snake populations are wild and one living in captivity, we would probably not choose to treat the intercepts as random. Rather, we would do so only after having accounted for another fixed effect coding for life in the wild versus in captivity.

The distribution that collects together a set of random effects is called a prior distribution by Bayesians. This may be slightly confusing, since it is not exactly the same kind of prior distribution as the distribution with which we describe our knowledge about each parameter of a model; this typically has known constants only. Instead, a prior governing a set of random effects itself has parameters that must be estimated and which therefore need priors themselves. Hence, the parameters of the random-effects distribution become hyperparameters and their priors become hyperpriors. Such a hierarchical scheme may include additional levels, so we might have hyper-hyperparameters and so forth. Of course, these names would soon become unwieldy, and indeed, are uninteresting, but it is important to recognize the common principle: as soon as we assume effects are random, we introduce a hierarchy of effects. The parameters at one level of the hierarchy can be seen as the realization of some probability distribution, the parameters of which may themselves be the realization of another probability distribution one level up. Hence, any model with random effects is intrinsically a hierarchical model; see Section 2.8.

Typically in ecology, random effects are continuous, and a normal distribution is assumed for them. However, random effects may also be discrete, for instance, in the state-space representation of survival models (Chapters 7–11), in binomial mixture models (Chapter 12), or in occupancy models (Chapter 13). The prior distribution for these discrete random effects is the Bernoulli and the Poisson.

We find the terminology surrounding random-effects models fairly confusing, so here we try to make some connections:

- Random effects always imply grouped effects; we model grouped parameters, batches, or sets of parameters (Gelman, 2005). We cannot declare a single effect as random.
- The factor that indexes group membership is called a random-effects factor or random factor for short, and the groups of a factor in general (fixed or random) are also called levels.
- The simplest random effects perhaps are intercepts or mean effects, and the associated models are sometimes called random-intercepts or varying-intercepts models (Gelman and Hill, 2007). However, we can just as well assume that a collection of slopes comes from a common distribution, leading to a random-coefficients or varying-slopes model (Gelman and Hill, 2007). Typically, random-coefficients models also contain random intercepts.
- We can also model variance parameters as random (Lee and Nelder, 2006).
- Models with random-effects factors are typically called random-effects models. When they also have fixed effects, they become mixed models or mixed-effects models. The following names are largely synonymous or at least overlap with the term, mixed model: variance-components model, hierarchical model, state-space model, latent-component, and latent-variable model.

4.1.3 Why Do We Treat Batches of Effects as Random?

There are many different motivations for random-effects modeling. This section lists some of them.

Scope of Inference

With random effects, the scope of inference of a study extends beyond the particular levels of a factor that appear in the study. In our snake example, this formally allows us to generalize to a larger number of populations, from which the three population studied were drawn from. Hence, we buy greater generality in treating our populations as representatives of a larger (statistical) population of populations that we *could* have studied and about which we would also like to learn something, for instance about how they

vary among each other. In truth, such a population of populations may be fairly hypothetical and hard to define in practice, yet it may be sensible to assume it exists. Of course, we can estimate the effects of the particular populations in our study. In addition, when random, we can also estimate or predict the effects of populations that are *not* in the studied sample but that belong to the same statistical population of populations. This is interesting for predicting a process into the future, for instance, when temporal random effects are included in a model, or for making similar extrapolations over space with spatial random effects.

Assessment of Variability

A random-effects model often focuses on the variability in a process. For instance, in a population viability analysis (Beissinger, 2002), the appropriate way to estimate temporal variance in a quantity like a survival probability is to model annual survival as a random effect (see Chapter 7). Actually, we can then estimate two kinds of variability: first, the variation among the levels in the population, that is, the population standard deviation (or variance); this is the parameter estimated in the model (e.g., σ_α^2 in the snake example). However, we can also estimate the variation among the observed levels in our study, that is, $\mathtt{sd}(\alpha_j)$ in our example. This is called the finite-population (or finite-sample) standard deviation. Gelman (2005) suggests the latter as a unified measure of the importance of a factor.

Partitioning of Variability

A random-effects factor codes for unstructured variability among the units in a group. For instance, it quantifies the variability in some process such as survival over a series of years. We may then be interested to explain that variability by covariates, for instance, a weather covariate measured every year. Covariate effects can be assessed by fitting a model first with a random-effects factor alone, for example, time, and second with a temporally varying covariate included. The proportional reduction in the time variance component is a measure for the proportion of the temporal variance in survival explained by that covariate (e.g., Grosbois et al., 2008; see also Section 7.4.3).

Modeling of Correlations among Parameters

Since we can assess the variation among the levels of one factor using random effects, we can also assess the covariation among pairs of levels of two factors using correlated random effects. For instance, we may model a correlation between intercepts and slopes (see Kéry, 2010) or between pairs of annual values of survival and fecundity (Cam et al., 2002) or juvenile and adult survival (see Section 7.6.2). Similarly, most modeling of spatial or temporal autocorrelation first declares random site or time effects, on which a correlation structure can then be imposed. For instance, a random

time effect at $t+1$ may be assumed to depend on the time effect at t in an autoregressive time series (Chapter 5). Similarly, spatial correlation may be modeled by assuming a correlation matrix for random site effects, with the correlation being a function of the distance between the sites (e.g., Royle et al., 2007b).

Accounting for All Random Processes in a Modeled System

Random-effects modeling results in a more honest accounting for the total uncertainty in a modeled system. Treating some components as random rather than as fixed recognizes the additional variability in, and hence uncertainty stemming from, these components. For instance, in our snake example, a replication of the study might have chosen different populations, with slightly different average mass, and by treating snake populations as random rather than fixed accounts for the fact that this sampling process introduces additional variability in our study, that is, additional uncertainty in our conclusions. If we estimate mass–length regression lines without accounting for this component of variation among populations and implicitly assume that our results are valid for other snake populations as well, our standard error for the regression lines will be too small. Thus, random effects account for the added uncertainty in the form of the sampling error incurred by sampling just some populations in a study, while many other, different, but similar and therefore exchangeable populations could have been selected instead.

Avoiding Pseudoreplication

Historically, one of the first motivations for random-effects modeling was the desire to account for inherent structure in many data sets, that is, to correct for intrinsic correlations and thereby avoid pseudoreplication (Hurlbert, 1984). For instance, in a study where plant height is measured for each of a sample of plants drawn from multiple populations, a random population effect will avoid committing pseudoreplication. The shared population effect in the measurement of two plants in the same population induces a correlation in their height. This correlation coefficient is sometimes called intraclass correlation; it is equal to $\sigma^2_{\text{pop}}/(\sigma^2_{\text{pop}}+\sigma^2_{\text{res}})$, where σ^2_{pop} is the population variance component, and σ^2_{res} the residual variance component among plants within populations. Similarly, the blocking structure in a designed experiment will be accounted for by random block effects.

Borrowing Strength

Random-effects modeling consists of constraining grouped parameters by a common distribution. Effects are no longer estimated independently from one another; rather, the estimate of each effect is influenced by all

members in the group. Thus, the groups share information. One also says that individual estimates "borrow strength from the ensemble". If the assumption of exchangeability holds, individual estimates will be improved (Link and Sauer, 1996; Sauer and Link, 2002; Welham et al., 2004). One consequence is that individual estimates are pulled in (shrunk) toward the grand mean, an effect called shrinkage. Effects that are estimated with less precision or are more extreme are pulled in more than the effects estimated with greater precision and that are more "average". This can be seen in the snake example, where the intercept of the more extreme red population is shrunk most. Shrinkage is greater when group effects (σ^2_{pop}) are smaller relative to the residual variation (σ^2_{res}), that is, when the intraclass correlation is small. With large sample size in each group (i.e., small σ^2_{res}) and highly variable groups (large σ^2_{pop}), hardly any shrinkage takes place. Random-effects estimates then become essentially identical to fixed-effects estimates. In fact, fixed-effects are a special case of random-effects estimates when the group variance component (σ^2_{pop}) is infinite.

Random Effects as a Compromise between Pooling and No Pooling of Batched Effects

Random-effects modeling can also be seen as an alternative to model selection: rather than making a hard decision about whether a factor should be in the model or out, we let the data (and the model) decide and allow a factor to be in the model in a fractional manner. Gelman and Hill (2007) argue that the random-effects assumption represents an intermediate between assuming all groups (levels) of a factor have an effect and assuming that none of them has an effect. Under the first assumption, all effects are independent and not pooled; this results in the classical fixed-effects estimates. Under the second assumption, the effects are absent or pooled, that is, we simply estimate the grand mean. In contrast, the random-effects assumption is equivalent to partial pooling of the effects, with the degree of pooling depending on the relative size of the among-group variation (σ^2_{pop}) and the within-group variation (σ^2_{res}) and on the precision of each individual group mean estimate.

Combining Information

Random-effects modeling is a natural way in which information may be combined over groups. For instance, results from different studies can be combined by assuming study random effects in a meta-analysis. The combined estimate is then automatically weighed by the information content of each study. Similarly, when combining the analysis of multiple data sets, each data set might be considered a sample from some larger hypothetical population of such data sets and modeled as a random effect.

4.1.4 Why Should We Ever Treat a Factor as Fixed?

So why do we not always treat all effects as random? There are several reasons.

First, the assumption of exchangeability may simply not hold and units in a group of effects may differ systematically. The assumption of a common prior distribution is then not sensible. "Borrowing strength" would backfire in these cases, by pulling extreme effects estimates toward the mean of the effects, in spite of them being *really* different and extreme.

Second, treating a factor with very few levels as random will result in very imprecise estimates of the hyperparameter. Hence, when factors have fewer than, say, 5–10 levels, they will rarely be treated as random (but see Gelman, 2005).

Third, random effects are computationally more expensive. We will see this many times with WinBUGS: as soon as we move from a fixed-effects formulation of a model to the corresponding random-effects version of the model, we need to run longer Markov chains to get convergence and run times will be increased, and sometimes greatly so. Similarly, it is often more difficult to get converging Markov chains in the analysis of random-effects versions of a model compared with the grouped effects when they are assumed fixed.

Fourth, another practical reason may be that many ecologists find random-effects models more difficult to understand. We believe that this has to do with the way in which random-effects models are presented in statistics classes at university, that is, within an ANOVA framework. To many, this is challenging and somewhat opaque. In contrast, within a linear model framework, with the models written in algebra, it becomes much more transparent what random effects are, as we have seen above. Another practical reason may be that historically, flexible software for modeling random effects has become widely available much later than software for modeling traditional fixed-effects models.

4.2 ACCOUNTING FOR OVERDISPERSION BY RANDOM EFFECTS-MODELING IN R AND WinBUGS

We introduce random effects in an application that may not be so familiar: as a way of accounting for overdispersion in count data. Overdispersion in Poisson or binomial responses denotes the situation that a response is more variable than prescribed by these two distributions where the variance is a function of the mean and thus not a free parameter. Overdispersion arises when important covariates are not in the model or when units are not independent, such as when nestlings are grouped in the same nests or plants within the same population.

Not accounting for this overdispersion would lead to too liberal tests and too short (e.g., 95% credible) intervals. In our example, we estimate one random effect for each observation. Thus, in a sense, we fit extra residuals in addition to the implicit residuals coming with the Poisson assumption and assume that the extra residuals are normal on the log link scale. More typical applications of random-effects modeling usually have effects that are shared by more than one observation (see Section 4.3 for random site and random year effects).

Overdispersion is very frequent when modeling counts; actually, it is the rule rather than the exception. The variance of counts modeled by a Poisson is not a free parameter, so when fitting a Poisson GLM with a function such as `glm()`, the residual deviance of a model should be about the same as the residual degrees of freedom. For example, look at the analysis of the (real) peregrine counts in Section 3.2.2:

```
[ ... ]
    Null deviance: 1591.395  on 39  degrees of freedom
Residual deviance:   76.448  on 36  degrees of freedom
[ ... ]
```

The residual deviance is more than twice as large as the residual degrees of freedom, a clear indication that the observations (i.e., the counts) are more variable than expected under the Poisson assumption. Since we have one observation per year, we may imagine the presence of unmodeled year effects. These can be modeled as coming from a normal distribution. The resulting model is also called the Poisson–log-normal (or PLN) model; see also chapter 15.1 in Gelman and Hill (2007), or Millar (2009). Other possibilities to account for overdispersion in R would be to use a negative binomial distribution or quasi-likelihood (Lee and Nelder, 2000). For the latter, see the family argument `quasipoisson` in the R function `glm()`.

As usual, it is enlightening to write down the model in algebra. In Section 3.2.2, we had assumed the following GLM for count C_i in year i:

1. Statistical distribution to describe the random part of the response:
Poisson

$$C_i \sim \text{Poisson}(\lambda_i)$$

2. Link function to transform the mean of the parameter so that it can be expressed as a linear function of covariates and random effects: the log

$$\log(\lambda_i) = \eta_i$$

3. Linear predictor to describe the systematic part of the response:

$$\eta_i = \alpha + \beta_1 * \text{year}_i + \beta_2 * \text{year}_i^2 + \beta_3 * \text{year}_i^3 \quad \text{(old linear predictor)}$$

So far, the model is the usual Poisson GLM. To convert it into a Poisson GLMM, we simply introduce into the linear predictor random effects ε_i. Hence, we replace component 3 with a new linear predictor:

$$\eta_i = \alpha + \beta_1 * \text{year}_i + \beta_2 * \text{year}_i^2 + \beta_3 * \text{year}_i^3 + \varepsilon_i \quad \text{(new linear predictor)}$$

The fourth component of the model is the distribution of the extra residuals:

4. Distribution of extra residuals (ε_i): Normal

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

We could have written the same model without the ε terms, but instead making the intercepts year specific and collecting them together in a normal distribution like the following: $\alpha_i \sim \text{Normal}(\mu, \sigma^2)$. This is simply a reparameterization of the same model.

This completes our algebraic description of the model, and we will recognize later how the model description in the BUGS language is almost a direct transcription of this algebraic description. Note that the random effects ε are year specific, so we may call them random year effects, but we should not confuse them with the three polynomial regression effects of year. Also, we may only think of overdispersion here as being caused by unmodeled year effects because we have a single observation per year.

4.2.1 Generation and Analysis of Simulated Data

We will now assemble random-effects counts and then break them down using mixed modeling in R and WinBUGS. This will illustrate again that inference from frequentist and Bayesian analyses will often be numerically very similar. Note that some versions of the lme4 package do not allow the fitting of the PLN model, so you will have to choose a version of the package that does (e.g., 2.8.1 or 2.12.1).

```
data.fn <- function(n = 40, alpha = 3.5576, beta1 = -0.0912, beta2 = 0.0091,
   beta3 = -0.00014, sd = 0.1){
    # n: Number of years
    # alpha, beta1, beta2, beta3: coefficients of a
    #    cubic polynomial of count on year
    # sd: standard deviation of normal distribution assumed for year
       effects

    # Generate values of time covariate
    year <- 1:n

    # First level of noise: generate random year effects
    eps <- rnorm(n = n, mean = 0, sd = sd)
```

```
# Signal (plus first level of noise): build up systematic part of the
   GLM and add the random year effects
log.expected.count <- alpha + beta1 * year + beta2 * year^2 + beta3 *
   year^3 + eps
expected.count <- exp(log.expected.count)

# Second level of noise: generate random part of the GLM: Poisson
   noise around expected counts
C <- rpois(n = n, lambda = expected.count)

# Plot simulated data
plot(year, C, type = "b", lwd = 2, main = "", las = 1, ylab = "Population
   size", xlab = "Year", ylim = c(0, 1.1*max(C)))
lines(year, expected.count, type = "l", lwd = 3, col = "red")

return(list(n = n, alpha = alpha, beta1 = beta1, beta2 = beta2, beta3 =
   beta3, year = year, sd = sd, expected.count = expected.count, C = C))
}
```


We generate one data set. The expected count now contains a random contribution from each year, so it is no longer a smooth line as in Chapter 3. We can also say that the response contains two random components now: one from the year and the other a common Poisson residual.

```
data <- data.fn()
```

In R, we use the function `lmer()` in the lme4 package to fit the model.

```
library(lme4)
yr <- factor(data$year)          # Create a factor year
glmm.fit <- lmer(C ~ (1 | yr) + year + I(year^2) + I(year^3), family =
   poisson, data = data)

Warning message:
In mer_finalize(ans) : false convergence (8)
```


We fail to get convergence, so we see that numerical challenges are not restricted to Bayesian computation. We manually standardize the year covariate and see whether this helps—it does:

```
mny <- mean(data$year)
sdy <- sd(data$year)
cov1 <- (data$year - mny) / sdy
cov2 <- cov1 * cov1
cov3 <- cov1 * cov1 * cov1
glmm.fit <- lmer(C ~ (1 | yr) + cov1 + cov2 + cov3, family = poisson, data =
   data)
glmm.fit
Generalized linear mixed model fit by the Laplace approximation
Formula: C ~ (1 | yr) + cov1 + cov2 + cov3
   Data: data
  AIC    BIC  logLik  deviance
   63  71.44   -26.5        53
```

```
Random effects:
  Groups Name        Variance    Std.Dev.
yr       (Intercept) 0.0059868 0.077374
Number of obs: 40, groups: yr, 40

Fixed effects:
             Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)   4.33482     0.03412   127.05    < 2e-16 ***
cov1          1.22411     0.05627    21.75    < 2e-16 ***
cov2          0.06023     0.02686     2.24     0.0250 *
cov3         -0.23495     0.02934    -8.01   1.17e-15 ***
[ ... ]
```

We form predictions and plot them (Fig. 4.2). If we wish, we can inspect the random-effects estimates by typing `ranef(glmm.fit)`. We could also have formed predictions by typing `fitted(glmm.fit)`, but doing this by hand enhances our understanding.

```
R.predictions <- exp(fixef(glmm.fit)[1] + fixef(glmm.fit)[2]*cov1 +
   fixef(glmm.fit)[3]*cov2 + fixef(glmm.fit)[4]*cov3 +
   unlist(ranef(glmm.fit)))
lines(data$year, R.predictions, col = "green", lwd = 2, type = "l")
```

Here is the WinBUGS solution. Remember that WinBUGS parameterizes the normal distribution in terms of the precision, that is, one over the variance.

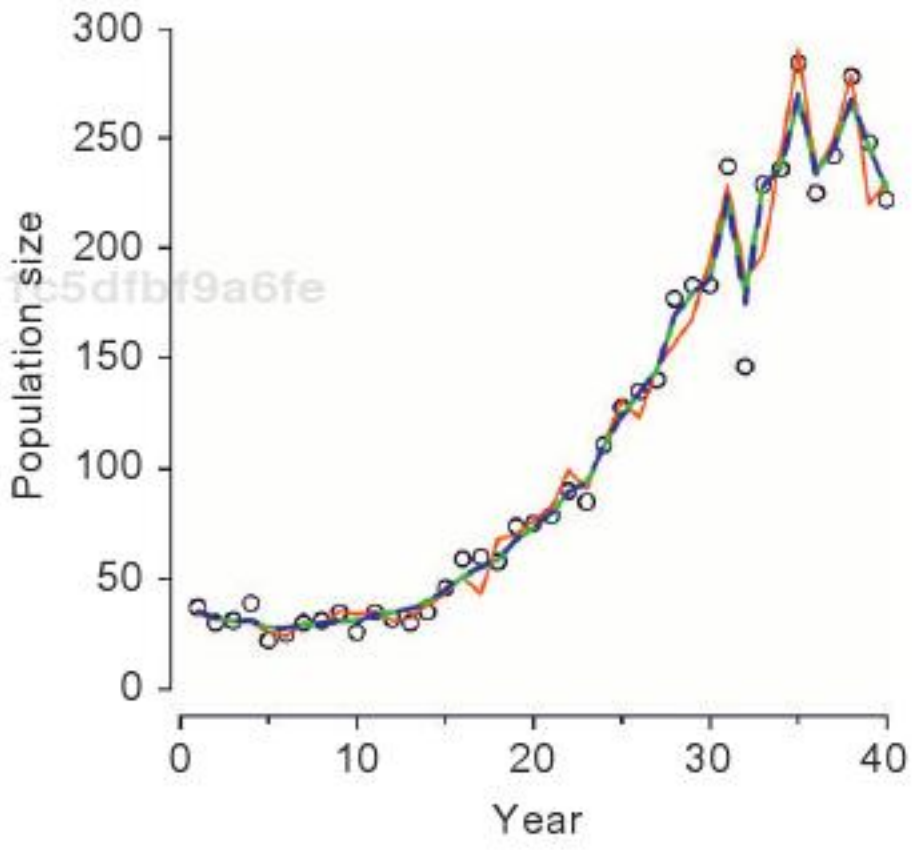

FIGURE 4.2 Simulated population size of peregrines in the French Jura over 40 years: expected population size (red), observed data (pair counts; black), estimated population trajectories from a frequentist (green, using REML in `lmer`), and a Bayesian analysis in WinBUGS (blue; posterior means) of a Poisson regression with cubic polynomial effects of year and year overdispersion. The R code to produce this figure slightly differs from the one shown in the book.

```
# Specify model in BUGS language
sink("GLMM_Poisson.txt")
cat("
model {

# Priors
alpha ~ dunif(-20, 20)
beta1 ~ dunif(-10, 10)
beta2 ~ dunif(-10, 10)
beta3 ~ dunif(-10, 10)
tau <- 1 / (sd*sd)
sd ~ dunif(0, 5)

# Likelihood: note key components of a GLM in one line each
for (i in 1:n){
    C[i] ~ dpois(lambda[i])           # 1. Distribution for random part
    log(lambda[i]) <- log.lambda[i]   # 2. Link function
    log.lambda[i] <- alpha + beta1 * year[i] + beta2 * pow(year[i],2) +
       beta3 * pow(year[i],3) + eps[i]  # 3. Linear predictor incl.
                                        random year effect
    eps[i] ~ dnorm(0, tau)        # 4. Definition of random effects dist
    }
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = data$C, n = length(data$C), year = cov1)

# Initial values
inits <- function() list(alpha = runif(1, -2, 2), beta1 = runif(1, -3, 3),
   sd = runif(1, 0,1))

# Parameters monitored
params <- c("alpha", "beta1", "beta2", "beta3", "lambda", "sd", "eps")

# MCMC settings
ni <- 30000
nt <- 10
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out<- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())
```

Here are two important comments. First, when calling the `bugs()` function, we will from now on use so-called positional matching of the arguments. That is, unlike in Chapter 3, where we wrote `bugs(data = win.data, inits = inits, ...)`, we will now let the position of some arguments determine their identity, so the first argument is data, the second is the inits function, and so on. This achieves more succinct code. Second, we will no longer formally comment on convergence in every example. In reality, we always check for convergence by looking at the plots in WinBUGS and

by inspecting the values of Rhat in the summary produced by the `bugs()` function, we just do not say so.

```
# Summarize posteriors
print(out, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
alpha	4.33	0.04	4.26	4.31	4.33	4.36	4.40	1.00	1800
beta1	1.22	0.06	1.11	1.18	1.22	1.26	1.35	1.00	450
beta2	0.06	0.03	0.01	0.04	0.06	0.08	0.12	1.00	1300
beta3	-0.23	0.03	-0.30	-0.25	-0.23	-0.21	-0.18	1.01	360
lambda[1]	35.51	4.20	28.04	32.55	35.23	38.10	44.84	1.00	1700
lambda[2]	32.02	3.53	25.48	29.49	31.93	34.43	39.20	1.00	1200
lambda[3]	30.86	3.13	25.12	28.75	30.84	32.82	37.37	1.00	2600
[...]									
lambda[38]	267.75	14.22	241.29	257.80	267.30	277.40	296.30	1.00	660
lambda[39]	246.04	13.55	220.20	236.67	245.80	255.22	273.00	1.00	3000
lambda[40]	226.47	13.47	200.90	216.90	226.55	235.30	252.90	1.00	3000
sd	0.09	0.02	0.05	0.07	0.09	0.10	0.14	1.01	340

We get estimates that are fairly similar to those from the frequentist analysis using maximum likelihood. We plot the predicted population trajectory into Fig. 4.2.

```
WinBUGS.predictions <- out$mean$lambda
lines(data$year, WinBUGS.predictions, col = "blue", lwd = 2, type = "l",
   lty = 2)
```

Hence, overdispersion in counts can be accounted for by adding random effects at the data level (here, random year effects ε_i), assuming a normal distribution on the scale of the linear predictor for them, and estimating the spread of that normal distribution. Accounting for overdispersion more adequately accounts for the full uncertainty in the modeled system. We can see this by comparing the standard errors of the regression estimates under a simple GLM with those obtained under the GLMM. We quickly fit the respective models in R.


```
glm.fit <- glm(C ~ cov1 + cov2 + cov3, family = poisson, data = data)
summary(glm.fit)
summary(glmm.fit)
```

The uncertainty of all regression estimates is slightly increased under the GLMM compared with the GLM. Thus, the GLMM propagates the additional uncertainty in the modeled system into the regression estimates as it should. See also exercise 1 in Section 4.5.

4.2.2 Analysis of Real Data

Now, let us repeat these analyses for the real data from the French Jura. We need to standardize year to avoid worries with convergence.

```
# Read data again
peregrine <- read.table("falcons.txt", header = TRUE)

yr <- factor(peregrine$Year)
mny <- mean(peregrine$Year)
sdy <- sd(peregrine$Year)
cov1 <- (peregrine$Year - mny) / sdy
cov2 <- cov1 * cov1
cov3 <- cov1 * cov1 * cov1
glmm <- lmer(peregrine$Pairs ~ (1 | yr) + cov1 + cov2 + cov3, family =
   poisson, data = peregrine)
glmm
Generalized linear mixed model fit by the Laplace approximation
Formula: peregrine$Pairs ~ (1 | yr) + cov1 + cov2 + cov3
    Data: peregrine
    AIC   BIC logLik deviance
  77.01 85.46 -33.51    67.01
Random effects:
 Groups  Name        Variance   Std.Dev.
 yr      (Intercept) 0.0098814  0.099405
Number of obs: 40, groups: yr, 40

Fixed effects:
             Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)   4.21205     0.03841   109.65   < 2e-16 ***
cov1          1.19085     0.06507    18.30   < 2e-16 ***
cov2          0.01720     0.02980     0.58     0.564
cov3         -0.27162     0.03361    -8.08   6.4e-16 ***
[ ... ]
```

We estimate an overdispersion standard deviation of about 0.1 (R also gives the square of that estimate, that is, the variance).

Next, we conduct the analysis in WinBUGS. We can reuse most ingredients of the analysis from Section 4.2.1. Do not forget to check convergence; if there is trouble, you have to increase the chain length and/or the burnin period.

```
# Bundle data
win.data <- list(C = peregrine$Pairs, n = length(peregrine$Pairs),
   year = cov1)

# MCMC settings (may have to adapt)
ni <- 30000
nt <- 10
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())
```

```
# Summarize posteriors
print(out, dig = 3)
                mean     sd    2.5%     25%     50%     75%   97.5%  Rhat n.eff
alpha          4.210  0.043   4.120   4.181   4.211   4.239   4.291 1.002  1700
beta1          1.200  0.078   1.057   1.147   1.197   1.250   1.361 1.005   420
beta2          0.017  0.033  -0.048  -0.004   0.016   0.038   0.083 1.002  1500
beta3         -0.275  0.040  -0.356  -0.302  -0.274  -0.247  -0.200 1.005   440
lambda[1]     34.252  4.364  26.549  31.130  34.010  37.010  43.451 1.002  1400
lambda[2]     35.796  4.640  27.859  32.487  35.510  38.590  46.121 1.001  3000
lambda[3]     32.141  3.800  25.450  29.490  31.850  34.540  40.401 1.001  3000
[ . . . ]
lambda[39]   180.358 12.146 157.600 171.775 180.050 188.600 205.100 1.001  3000
lambda[40]   176.702 12.751 152.597 168.100 176.300 185.300 202.402 1.001  3000
sd             0.117  0.032   0.059   0.095   0.116   0.136   0.182 1.002  3000
[ . . . ]
```

As usual, we get similar estimates for both mean and variance parameters in R and WinBUGS. It is instructive to compare the posterior summaries from Poisson GLMM with those under the simple Poisson GLM in Section 3.2.2: you will see that the estimates of the mean parameters (alpha, beta1, beta2, and beta3) are not that different under both models. However, the uncertainty around them (posterior sd) is greater under the mixed Poisson model. Similarly, the 95% CRI are wider, and also the uncertainty around the lambda components is larger. Hence, our estimates properly account for the added uncertainty introduced by the second component of variation in the modeled system, the random year effects. Without accounting for the extra-Poisson variability in the counts, we would have underestimated parameter uncertainty.

4.3 MIXED MODELS WITH RANDOM EFFECTS FOR VARIABILITY AMONG GROUPS (SITE AND YEAR EFFECTS)

For the remainder of this chapter, we look at models for population counts that are indexed by both space and time, rather than only by time as in the preceding examples in this chapter. We will model counts $C_{i,j}$ made at time i and site j as a function of cubic polynomials of year and start by assuming a Poisson distribution with expected count $\lambda_{i,j}$:

$$C_{i,j} \sim \text{Poisson}(\lambda_{i,j}).$$

To model spatiotemporal structure in the expected counts, we apply the usual log link function and express the log (expected count) as a function of covariates in a linear way:

$$\log(\lambda_{i,j}) = \alpha_j + \sum_k \beta_p * X_i^p + \varepsilon_i$$

This equation describes a log-linear multiple regression of the expected count on an additive function of $k = 3$ time covariates X, that is, a fixed-effects Poisson GLM, with an added residual term ε_i. Now, look at the indices of the parameters: the slope parameters (betas) do not vary by time or by among sites (they are not indexed by i or by j): we estimate a single set of beta for all sites and times. However, the intercept α is indexed by site (j) and so codes for site effects, and the residual ε is indexed by time (i) and hence codes for time effects in addition to those of the covariate X.

The model, as written so far, is a fixed-effects model. The GLMMs considered in the rest of this chapter extend the fixed-effects Poisson GLM by adding two (or more) distributional assumptions to the equation above, for instance

$$\alpha_j \sim \text{Normal}(\mu, \sigma_\alpha^2) \text{ and}$$

$$\varepsilon_i \sim \text{Normal}(0, \sigma_\varepsilon^2).$$

This allows for stochastic site differences in the level of the population trajectory, that is, in the intercepts α_j, and for stochastic year effects ε_i common to all sites. Note that we could also estimate random year-by-site noise, say $\varepsilon_{i,j}$, but we will assume here that our current model is sufficiently flexible. To describe each set of parameters, α_j and ε_i, we will estimate one (hyper)parameter that expresses the magnitude of the variability among sites (σ_α^2) and another for the variability among years (σ_ε^2).

Note that a simple reparameterization of this model would consist in "pulling out" the mean site effect, μ, into an overall intercept. The site effects would then come from a mean-zero normal distribution rather than one with mean μ.

$$\log(\lambda_{i,j}) = \mu + \sum_k \beta_p * X_i^p + \alpha_j + \varepsilon_i$$

$$\alpha_j \sim \text{Normal}(0, \sigma_\alpha^2)$$

$$\varepsilon_i \sim \text{Normal}(0, \sigma_\varepsilon^2)$$

Models of this kind have become sort of standard for analyzing population counts (Link and Sauer, 2002; Ver Hoef and Jansen, 2007; Cressie et al., 2009, among many others). They are often called hierarchical models, since they contain a hierarchy of effects, and we can think of a nested relationship among the parameters in the model. However, the term, hierarchical model, by itself is about as informative about the actual model used as it would be to say that I am using a four-wheeled vehicle for locomotion; there are golf carts, quads, Smart cars, Volkswagen, and

Mercedes, for instance, and they all have four wheels. Similarly, plenty of statistical models applied in ecology have at least one set of random effects (beyond the residual) and therefore represent a hierarchical model. Hence, the term is not very informative about the particular model adopted for inference about an animal population.

4.3.1 Generation and Analysis of Simulated Data

For data simulation, we adapt the data generation function from Section 4.2.1 to several parallel time series of counts and to allow for random site and random year effects. We assume that there are site effects on the intercept of the trajectory only, not the slope parameters. So, we might think that this function generates replicate trajectories of counts of the peregrine population in the French Jura. We consider balanced data for convenience only; the models work also if there are missing values in the year-by-site data.

```
data.fn <- function(nsite = 5, nyear = 40, alpha = 4.18456, beta1 =
   1.90672, beta2 = 0.10852, beta3 = -1.17121, sd.site = 0.5, sd.year =
   0.2){
    # nsite: Number of populations
    # nyear: Number of years
    # alpha, beta1, beta2, beta3: cubic polynomial coefficients of year
    # sd.site: standard deviation of the normal distribution assumed for
       the population intercepts alpha
    # sd.year: standard deviation of the normal distribution assumed for
       the year effects
    # We standardize the year covariate so that it runs from about -1 to 1

    # Generate data structure to hold counts and log(lambda)
    C <- log.expected.count <- array(NA, dim = c(nyear, nsite))

    # Generate covariate values
    year <- 1:nyear
    yr <- (year-20)/20  # Standardize
    site <- 1:nsite

    # Draw two sets of random effects from their respective distribution
    alpha.site <- rnorm(n = nsite, mean = alpha, sd = sd.site)
    eps.year <- rnorm(n = nyear, mean = 0, sd = sd.year)

    # Loop over populations
    for (j in 1:nsite){

      # Signal (plus first level of noise): build up systematic part of
         the GLM including random site and year effects
      log.expected.count[,j] <- alpha.site[j] + beta1 * yr + beta2 * yr^2
         + beta3 * yr^3 + eps.year
      expected.count <- exp(log.expected.count[,j])
```

```
      # Second level of noise: generate random part of the GLM: Poisson
      noise around expected counts
C[,j] <- rpois(n = nyear, lambda = expected.count)
      }

# Plot simulated data
matplot(year, C, type = "l", lty = 1, lwd = 2, main = "", las = 1, ylab =
   "Population size", xlab = "Year")

return(list(nsite = nsite, nyear = nyear, alpha.site = alpha.site,
   beta1 = beta1, beta2 = beta2, beta3 = beta3, year = year, sd.site = sd.
   site, sd.year = sd.year, expected.count = expected.count, C = C))
    }
```

We generate one very large data set and plot it (Fig. 4.3). Simulating 100 populations will result in a very large BUGS run time, and you may want to choose fewer sites to reduce the run time in your analysis, for example, nsite = 10 will result in a BUGS run time (BRT) of about 12 min.

```
data <- data.fn(nsite = 100, nyear = 40, sd.site = 0.3, sd.year = 0.2)
```

We analyze using BUGS.

```
# Specify model in BUGS language
sink("GLMM_Poisson.txt")
cat("
model {
```

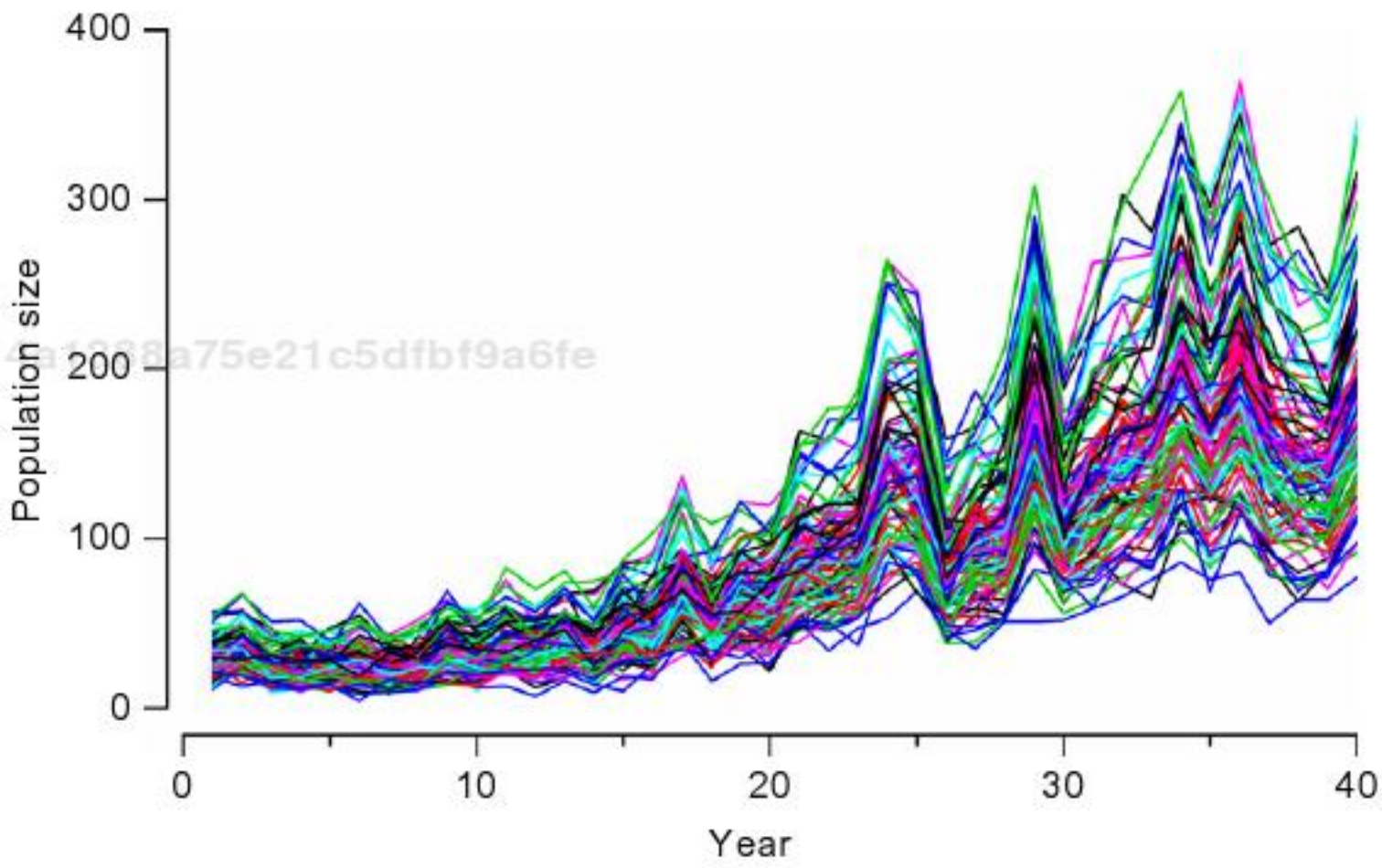

FIGURE 4.3 One hundred replicate population trajectories with additive random site and year effects. We emphasize how a single stochastic process, here, the `data.fn()` function and in an ecologist's daily life, nature, can produce spectacularly different outcomes. In nature, we typically only ever see a single outcome of the stochastic process from which we want to infer the characteristics of the latter—a difficult challenge.

```
# Priors
for (j in 1:nsite){
    alpha[j] ~ dnorm(mu, tau.alpha)          # 4. Random site effects
    }
mu ~ dnorm(0, 0.01)                          # Hyperparameter 1
tau.alpha <- 1 / (sd.alpha*sd.alpha)         # Hyperparameter 2
sd.alpha ~ dunif(0, 2)
for (p in 1:3){
    beta[p] ~ dnorm(0, 0.01)
    }

tau.year <- 1 / (sd.year*sd.year)
sd.year ~ dunif(0, 1)                        # Hyperparameter 3

# Likelihood
for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.year)              # 4. Random year effects
    for (j in 1:nsite){
       C[i,j] ~ dpois(lambda[i,j])           # 1. Distribution for random
                                                part
       lambda[i,j] <- exp(log.lambda[i,j]) # 2. Link function
       log.lambda[i,j] <- alpha[j] + beta[1] * year[i] + beta[2] *
          pow(year[i],2) + beta[3] * pow(year[i],3) + eps[i] # 3. Linear
          predictor including random site and random year effects
       } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = data$C, nsite = ncol(data$C), nyear = nrow(data$C),
   year = (data$year-20) / 20) # Note year standardized

# Initial values
inits <- function() list(mu = runif(1, 0, 2), alpha = runif(data$nsite,
   -1, 1), beta = runif(3, -1, 1), sd.alpha = runif(1, 0, 0.1),
   sd.year = runif(1, 0, 0.1))

# Parameters monitored (may want to add "lambda")
params <- c("mu", "alpha", "beta", "sd.alpha", "sd.year")

# MCMC settings (may have to adapt)
ni <- 100000
nt <- 50
nb <- 50000
nc <- 3

# Call WinBUGS from R (BRT 98 min)
out <- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out, dig = 3)
```

Estimation in more complex mixed models can be challenging. Estimating variance parameters is always more difficult than estimating fixed-effects parameters. This is manifest in slower convergence rates of the Markov chains, longer run times or trouble in getting convergence at all. Problems are compounded by small sample sizes. For the estimation of a variance parameter, the relevant sample size is the number of groups among which we want to estimate the variability in some parameter. Estimating variance parameters with fewer than 5–10 groups can be particularly difficult.

4.3.2 Analysis of Real Data Set

We analyze data from the Swiss breeding bird survey MHB ("Monitoring Häufige Brutvögel"; Schmid et al., 2004), a scheme launched in 1999. During each breeding season, a systematic sample of about 300 1-km^2 quadrats is surveyed 2–3 times using the territory mapping method. This produces a count of putative bird territories at each site and year. We will use a sample of data from 235 sites over 9 years and model the annual territory counts of the coal tit (Fig. 4.4). We are interested in whether there is an overall population trend in Swiss coal tits and in the variation of counts among sites, years, and observers. In addition, we would like to see whether observers count fewer birds during their first year of survey in a particular quadrat.

FIGURE 4.4 Coal tit (*Parus ater*), Finland, 2007. (Photograph by M. Varesvuo.)

Typically, each observer surveys only a small number of quadrats per year, but surveys the same quadrat(s) over a series of years.

We will fit a sequence of models starting with fixed effects only and then move on to including random effects. The comparison of analogous fixed- and random-effects models will be illuminating for your understanding of random effects. The most complex model entertained for count $C_{i,j}$ in year i at site j will be as follows:

$$C_{i,j} \sim \text{Poisson}(\lambda_{i,j}) \qquad \text{Data Distribution}$$

$$\log(\lambda_{i,j}) = \alpha_j + \beta_1 * \text{year}_i + \beta_2 * F_{i,j} + \delta_i + \gamma_{k(i,j)} \qquad \text{Link Function and Linear Predictor}$$

$$\alpha_j \sim \text{Normal}(\mu, \sigma_\alpha^2) \qquad \text{Random Site Effects}$$

$$\delta_i \sim \text{Normal}(0, \sigma_\delta^2) \qquad \text{Random Year Effects}$$

$$\gamma_{k(i,j)} \sim \text{Normal}(0, \sigma_\gamma^2) \qquad \text{Random Observer Effects}$$

Thus, the log expected count, $\log(\lambda_{i,j})$, is a linear function of random site effects α_j, random year effects δ_i, and random observer effects $\gamma_{k(i,j)}$. In addition, there is a slope on year of magnitude β_1 (i.e., a trend) and a change in the expected count of magnitude β_2 if the indicator of first-year-of-service $F_{i,j}$ is equal to 1. The three sets of random effects are assumed to be drawn from independent normal distributions whose variances (or standard deviations) we estimate along with a grand mean μ.

```
# Read in the tit data and have a look at them
tits <- read.table("tits.txt", header = TRUE)
str(tits)
```

We have two environmental covariates (elevation and forest cover), annual territory counts 1999–2007 (y1999, etc.), an observer code (obs1999, etc., scaled to protect true observer ID), and finally a set of indicators for years when an observer surveyed a quadrat for the first time (first1999, etc.). Variable first1999 contains 1's only (except for some NAs), since MHB was launched in 1999. When modeling first-time observer effects, we may want to discard the 1999 data; otherwise the year 1999 effect is confounded with the first-time observer effect.

We collect counts, observer code, and first-time observer indicator variables in separate matrices and then plot the counts (Fig. 4.5; note use of the transposition function `t()`).

```
C <- as.matrix(tits[5:13])
obs <- as.matrix(tits[14:22])
first <- as.matrix(tits[23:31])

matplot(1999:2007, t(C), type = "l", lty = 1, lwd = 2, main = "",
   las = 1, ylab = "Territory counts", xlab = "Year", ylim = c(0, 80),
   frame = FALSE)
```

From Fig. 4.5, it appears that the Swiss coal tit metapopulation fluctuates around some fairly constant level. We will thus consider only a linear

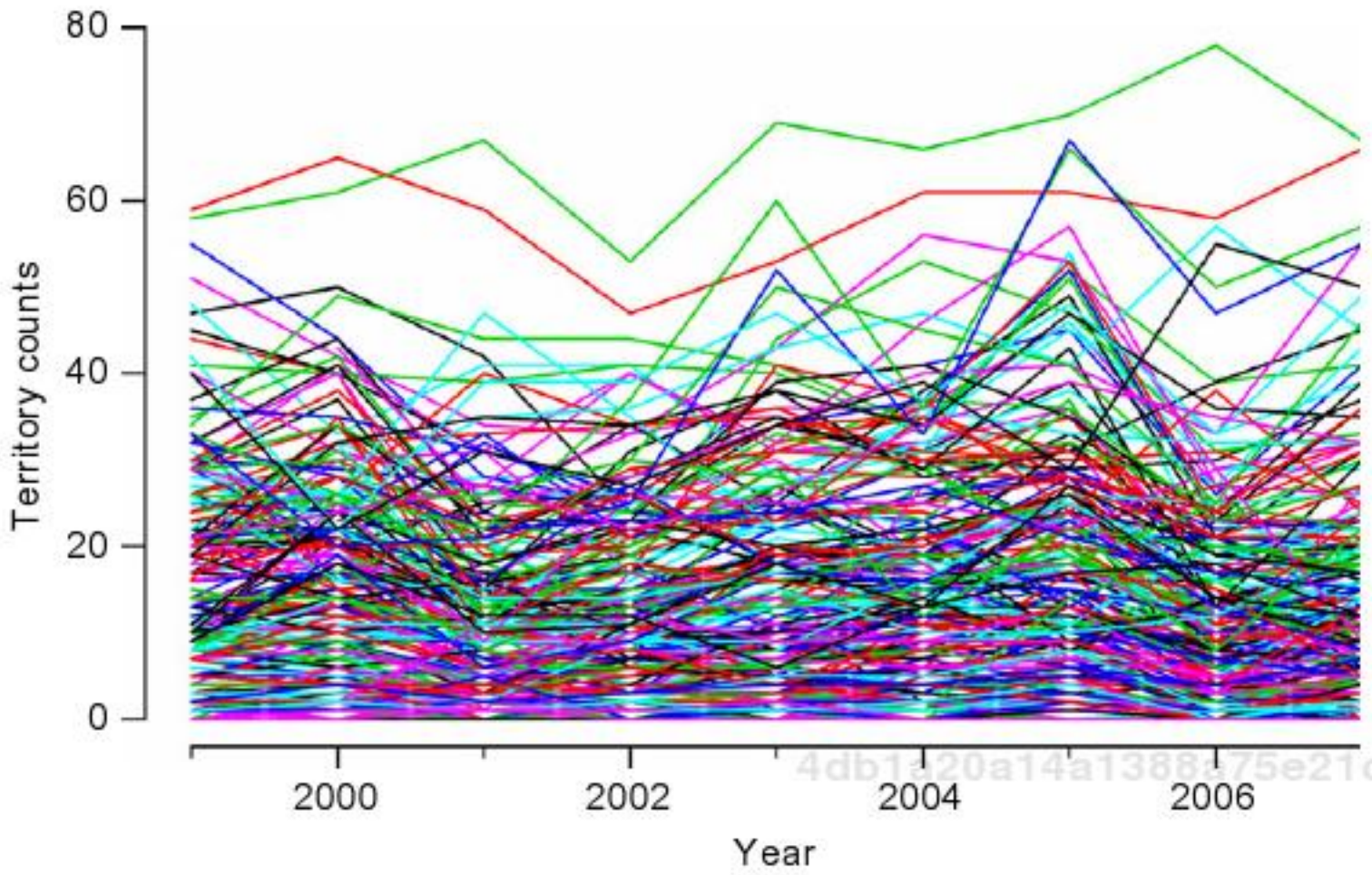

FIGURE 4.5 Territory counts of coal tits in 235-MHB quadrats in the Swiss breeding bird survey.

regression of counts on time rather than a higher order polynomial as for the French peregrines. However, we will try to explain variation among years and among sites by observer identity and first-year observer effects.

```
table(obs)
length(table(obs))
```

A total of 271 observers served from 1999 to 2007 and surveyed a fairly variable number of site years.

```
apply(first, 2, sum, na.rm = TRUE)
first1999 first2000 first2001 first2002 first2003 first2004 first2005
  first2006 first2007
      176       37       22       25       40       31       25       33       24
```

After 1999, about 10%–20% of observers are new on their quadrat each year. To model observer ID as a factor in WinBUGS, we need to recode it to be continuous.

```
a <- as.numeric(levels(factor(obs)))      # All the levels, numeric
newobs <- obs                             # Gets ObsID from 1:271
for (j in 1:length(a)){newobs[which(obs==a[j])] <- j }
table(newobs)
```

We also need to fill up any missing values in the explanatory variates in WinBUGS (or else we have to specify priors for them). Unfortunately, there are two kinds of missing values in the observer ID data: one for site years when a quadrat was not surveyed (these correspond to NAs in the counts matrix C) and another when a quadrat was surveyed, but the observer

identity was not recorded. We impute some values in them, for example, 272 for missing observer identity and a zero in the first-year observer indicator variable. When transforming all NAs in the observer ID matrix to 272, we lump all unknown observers into a single observer identity. Given the large number of observers, this should hardly have an effect on the inference.

```
newobs[is.na(newobs)] <- 272
table(newobs)
first[is.na(first)] <- 0
table(first)
```

We are now ready to model these counts. Our strategy will be to start with the simplest possible model, with an intercept only, and then gradually build in more complexity. More specifically, we will first fit some fixed-effects models and only then we add the additional random-effects assumptions about some sets of previous fixed effects. This modification should clarify the exact meaning of the random-effects assumption. We will look at the deviance information criterion (DIC) as long as all effects in the model are fixed, but abandon it when we move to random effects, since the use of the standard DIC seems to be problematic then (Millar, 2009).

Null or Intercept-Only Model

This model has a constant expected count throughout space and time.

```
# Specify model in BUGS language
sink("GLM0.txt")
cat("
model {

# Prior
alpha ~ dnorm(0, 0.01)      # log(mean count)

# Likelihood
for (i in 1:nyear){
   for (j in 1:nsite){
      C[i,j] ~ dpois(lambda[i,j])
      lambda[i,j] <- exp(log.lambda[i,j])
      log.lambda[i,j] <- alpha
      }   #j
   }   #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values
inits <- function() list(alpha = runif(1, -10, 10))
```

```
# Parameters monitored
params <- c("alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out0 <- bugs(win.data, inits, params, "GLM0.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out0, dig = 3)
              mean    sd      2.5%       25%       50%       75%     97.5% Rhat n.eff
alpha        2.668 0.006     2.656     2.664     2.667     2.672     2.679    1  1500
deviance 32853.240 4.724 32850.000 32850.000 32850.000 32860.000 32860.000    1  1500

pD = 11.2 and DIC = 32864.4 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).
```

This cannot be a very good model, but it's a start. It says that the mean observed density of tits per 1 km^2 is exp(2.67) = 14.4. Next, we add in fixed site effects.

Fixed Site Effects

This model is essentially a one-way ANOVA, except that it is for a Poisson rather than a normal response.

```
# Specify model in BUGS language
sink("GLM1.txt")
cat("
model {

# Priors
for (j in 1:nsite){
    alpha[j] ~ dnorm(0, 0.01)      # Site effects
    }

# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- alpha[j]
        }   #j
    }   #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))
```

```
# Initial values (not required for all)
inits <- function() list(alpha = runif(235, -1, 1))

# Parameters monitored
params <- c("alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out1 <- bugs(win.data, inits, params, "GLM1.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out1, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
alpha[1]	-0.17	0.36	-0.91	-0.40	-0.16	0.08	0.47	1.00	1500
alpha[2]	-0.49	0.41	-1.36	-0.76	-0.45	-0.19	0.22	1.00	1200
alpha[3]	2.46	0.10	2.26	2.40	2.46	2.53	2.65	1.00	980
[...]									
alpha[233]	3.68	0.05	3.57	3.64	3.68	3.71	3.78	1.00	1500
alpha[234]	3.36	0.06	3.24	3.32	3.36	3.40	3.48	1.00	1500
alpha[235]	3.59	0.06	3.48	3.56	3.59	3.63	3.70	1.00	490
deviance	11688.71	21.82	11650.00	11670.00	11690.00	11700.00	11730.00	1.00	1500

```
pD = 238.3 and DIC = 11927 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).
```

Fitting site effects has greatly improved the fit of the model; the deviance has come down by half and so has the DIC. Next, we also add fixed year effects.

Fixed Site and Fixed Year Effects

This model is a two-way, main-effects ANOVA for a Poisson response. Note how, in the inits function, we must not give an initial value for the first level of the year effects factor, which we have to set to zero to avoid overparameterization.

```
# Specify model in BUGS language
sink("GLM2.txt")
cat("
model {

# Priors
for (j in 1:nsite){          # Site effects
    alpha[j] ~ dnorm(0, 0.01)
    }
for (i in 2:nyear){          # nyear-1 year effects
    eps[i] ~ dnorm(0, 0.01)
    }
eps[1] <- 0                  # Aliased
```

```
# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- alpha[j] + eps[i]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values
inits <- function() list(alpha = runif(235, -1, 1), eps = c(NA, runif(8,
   -1, 1)))

# Parameters monitored
params <- c("alpha", "eps")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out2 <- bugs(win.data, inits, params, "GLM2.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out2, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
alpha[1]	-0.18	0.37	-0.96	-0.40	-0.16	0.07	0.49	1.00	1500
alpha[2]	-0.50	0.43	-1.37	-0.78	-0.47	-0.20	0.26	1.00	1500
alpha[3]	2.46	0.10	2.25	2.39	2.46	2.53	2.65	1.00	1000
[...]									
alpha[233]	3.67	0.06	3.56	3.64	3.67	3.71	3.78	1.00	920
alpha[234]	3.36	0.06	3.23	3.32	3.36	3.40	3.48	1.00	460
alpha[235]	3.59	0.06	3.47	3.55	3.59	3.63	3.69	1.00	1500
eps[2]	0.08	0.03	0.03	0.06	0.08	0.09	0.13	1.02	85
eps[3]	-0.16	0.03	-0.21	-0.18	-0.16	-0.14	-0.10	1.03	69
eps[4]	-0.11	0.03	-0.15	-0.12	-0.11	-0.09	-0.05	1.04	59
eps[5]	0.06	0.03	0.01	0.04	0.06	0.07	0.11	1.02	97
eps[6]	0.10	0.02	0.05	0.08	0.10	0.12	0.15	1.03	76
eps[7]	0.22	0.02	0.18	0.21	0.22	0.24	0.27	1.02	110
eps[8]	-0.16	0.03	-0.21	-0.18	-0.16	-0.14	-0.11	1.02	98
eps[9]	-0.07	0.03	-0.12	-0.08	-0.07	-0.05	-0.02	1.02	94
deviance	11237.79	22.12	11200.00	11220.00	11240.00	11250.00	11280.00	1.00	1500

```
pD = 244.7 and DIC = 11482.4 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).
```

With the introduction of year effects, the deviance and the DIC have gone down quite a bit again, so there seem to be annual population fluctuations. Now, we increase the model complexity by specifying random instead of fixed effects. Watch how simple the move from fixed to random effects is when a model is defined in the BUGS language. Essentially, all that is required is a common distribution to be assumed for a set of grouped effects with hyperparameters that are going to be estimated and that therefore need hyperpriors in turn. We start with the random site model.

Random Site Effects (No Year Effects)

The linear predictor of this model is that of a one-way, random-effects ANOVA.


```
# Specify model in BUGS language
sink("GLMM1.txt")
cat("
model {

# Priors
for (j in 1:nsite){
    alpha[j] ~ dnorm(mu.alpha, tau.alpha)    # Random site effects
    }
mu.alpha ~ dnorm(0, 0.01)
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- alpha[j]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values
inits <- function() list(mu.alpha = runif(1, 2, 3))

# Parameters monitored
params <- c("alpha", "mu.alpha", "sd.alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3
```

```
# Call WinBUGS from R (BRT < 1 min)
out3 <- bugs(win.data, inits, params, "GLMM1.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out3, dig = 2)
              mean    sd     2.5%      25%      50%      75%    97.5% Rhat n.eff
alpha[1]     -0.03  0.33    -0.69    -0.25    -0.02     0.21     0.57 1.00  1500
alpha[2]     -0.26  0.36    -1.03    -0.49    -0.24     0.00     0.40 1.00  1500
alpha[3]      2.46  0.11     2.25     2.39     2.46     2.52     2.67 1.00  1500
[ . . . ]
alpha[233]    3.68  0.05     3.57     3.64     3.67     3.71     3.78 1.00  1500
alpha[234]    3.36  0.06     3.24     3.32     3.36     3.40     3.48 1.00  1500
alpha[235]    3.59  0.06     3.48     3.55     3.59     3.63     3.69 1.00   650
mu.alpha      2.09  0.09     1.93     2.03     2.09     2.15     2.27 1.00  1500
sd.alpha      1.33  0.06     1.21     1.28     1.32     1.37     1.46 1.00  1500
deviance 11692.91 22.38 11650.00 11680.00 11690.00 11710.00 11740.00 1.00   650
```

The mass of the posterior distribution of the standard deviation of the random site effects is concentrated well away from zero. This confirms our previous conclusion that sites differ substantially in their expected counts of coal tits. We next add a set of random year effects and in addition reparameterize the model to have a single grand mean. Each set of random effects is then centered around that grand mean, which helps convergence.

Random Site and Random Year Effects

This linear model corresponds to a main-effects ANOVA with two random factors. In comparison to the fixed-effects counterpart of this model (GLM 2), we no longer need to constrain one effect of one factor to zero to avoid overparameterization. The borrowing strength among parameters within the same random-effects factor ensures that all can be estimated.


```
# Specify model in BUGS language
sink("GLMM2.txt")
cat("
model {
# Priors
mu ~ dnorm(0, 0.01)                          # Grand mean
for (j in 1:nsite){
    alpha[j] ~ dnorm(0, tau.alpha)           # Random site effects
    }
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.eps)               # Random year effects
    }
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 3)
```

```
# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- mu + alpha[j] + eps[i]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values (not required for all)
inits <- function() list(mu = runif(1, 0, 4), alpha = runif(235, -2, 2),
   eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "alpha", "eps", "sd.alpha", "sd.eps")

# MCMC settings
ni <- 6000
nt <- 5
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
out4 <- bugs(win.data, inits, params, "GLMM2.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out4, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mu	2.09	0.10	1.89	2.03	2.10	2.16	2.29	1.01	860
alpha[1]	-2.13	0.34	-2.83	-2.35	-2.12	-1.90	-1.51	1.00	3000
alpha[2]	-2.36	0.37	-3.14	-2.60	-2.35	-2.11	-1.68	1.00	2100
alpha[3]	0.36	0.13	0.10	0.27	0.36	0.45	0.62	1.00	590
[...]									
alpha[233]	1.57	0.10	1.38	1.51	1.57	1.64	1.76	1.01	260
alpha[234]	1.26	0.10	1.05	1.19	1.26	1.33	1.46	1.00	770
alpha[235]	1.49	0.10	1.30	1.42	1.49	1.55	1.68	1.01	310
eps[1]	0.00	0.06	-0.12	-0.03	0.00	0.04	0.12	1.03	88
eps[2]	0.08	0.06	-0.04	0.04	0.08	0.11	0.19	1.03	86
eps[3]	-0.16	0.06	-0.27	-0.19	-0.15	-0.12	-0.04	1.03	93
eps[4]	-0.10	0.06	-0.22	-0.14	-0.10	-0.07	0.01	1.04	67
eps[5]	0.06	0.06	-0.06	0.03	0.06	0.10	0.17	1.03	89
eps[6]	0.10	0.06	-0.02	0.06	0.10	0.13	0.21	1.03	89
eps[7]	0.22	0.06	0.11	0.19	0.22	0.26	0.34	1.05	77
eps[8]	-0.16	0.06	-0.28	-0.19	-0.16	-0.12	-0.05	1.03	83
eps[9]	-0.06	0.06	-0.18	-0.10	-0.06	-0.03	0.05	1.03	84
sd.alpha	1.33	0.07	1.21	1.28	1.33	1.37	1.47	1.00	3000
sd.eps	0.15	0.05	0.09	0.12	0.14	0.18	0.28	1.00	1100
deviance	11240.99	22.57	11200.00	11230.00	11240.00	11260.00	11290.00	1.00	3000

The year effects do not seem to be very large; at any rate, they (`sd.eps`) are much smaller than the variation of counts among sites (`sd.alpha`). Next, we add a fixed first-year observer effect.

Random Site and Random Year Effects and First-Year Fixed Observer Effect

This model is analogous to a three-way ANOVA with two factors random and one fixed and all of them acting in an additive way.

```
# Specify model in BUGS language
sink("GLMM3.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                    # Overall mean
beta2 ~ dnorm(0, 0.01)                 # First-year observer effect

for (j in 1:nsite){
    alpha[j] ~ dnorm(0, tau.alpha)     # Random site effects
    }
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)
for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.eps)         # Random year effects
    }
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 5)

# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- mu + beta2 * first[i,j] + alpha[j] + eps[i]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), first = t
   (first))

# Initial values
inits <- function() list(mu = runif(1, 0, 4), beta2 = runif(1, -1, 1),
   alpha = runif(235, -2, 2), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "beta2", "alpha", "eps", "sd.alpha", "sd.eps")

# MCMC settings
ni <- 6000
```

```
nt <- 5
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
out5 <- bugs(win.data, inits, params, "GLMM3.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out5, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mu	2.09	0.09	1.92	2.02	2.09	2.16	2.27	1.06	37
beta2	0.00	0.02	-0.04	-0.02	0.00	0.01	0.03	1.00	2800
alpha[1]	-2.12	0.33	-2.80	-2.35	-2.11	-1.88	-1.53	1.00	460
alpha[2]	-2.36	0.37	-3.15	-2.59	-2.34	-2.11	-1.69	1.00	1900
alpha[3]	0.36	0.14	0.09	0.27	0.36	0.46	0.62	1.02	140
[...]									
alpha[233]	1.58	0.10	1.38	1.51	1.58	1.65	1.77	1.03	68
alpha[234]	1.26	0.11	1.06	1.19	1.26	1.34	1.47	1.03	81
alpha[235]	1.49	0.10	1.29	1.42	1.50	1.56	1.70	1.03	74
eps[1]	0.01	0.06	-0.10	-0.03	0.00	0.04	0.14	1.05	74
eps[2]	0.08	0.06	-0.02	0.04	0.08	0.11	0.20	1.06	59
eps[3]	-0.15	0.06	-0.26	-0.19	-0.15	-0.12	-0.03	1.05	67
eps[4]	-0.10	0.06	-0.21	-0.14	-0.10	-0.07	0.02	1.05	71
eps[5]	0.06	0.06	-0.04	0.03	0.06	0.09	0.19	1.06	62
eps[6]	0.10	0.06	0.00	0.07	0.10	0.13	0.23	1.06	62
eps[7]	0.22	0.06	0.12	0.19	0.22	0.25	0.35	1.05	66
eps[8]	-0.16	0.06	-0.26	-0.19	-0.16	-0.12	-0.03	1.05	64
eps[9]	-0.06	0.06	-0.17	-0.10	-0.06	-0.03	0.06	1.05	66
sd.alpha	1.33	0.07	1.20	1.28	1.32	1.37	1.47	1.00	3000
sd.eps	0.16	0.05	0.09	0.12	0.14	0.18	0.29	1.00	1100
deviance	11240.86	22.54	11200.00	11230.00	11240.00	11250.00	11280.00	1.00	1100

`Beta2` is very nearly zero; hence, there is no evidence for a first-year observer effect. Next, we also add an overall linear time trend.

Random Site and Random Year Effects, First-Year Fixed Observer Effect, and Overall Linear Time Trend

The linear part of this model corresponds to an analysis of covariance (ANCOVA), with, in addition to the three factors of the previous model, an added effect of one continuous covariate.

```
# Specify model in BUGS language
sink("GLMM4.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                    # Overall intercept
beta1 ~ dnorm(0, 0.01)                 # Overall trend
beta2 ~ dnorm(0, 0.01)                 # First-year observer effect
```

```
for (j in 1:nsite){
    alpha[j] ~ dnorm(0, tau.alpha)     # Random site effects
    }
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.eps)         # Random year effects
    }
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 3)

# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- mu + beta1 * year[i] + beta2 * first[i,j] +
   alpha[j] + eps[i]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), first = t
   (first), year = ((1:9)-5) / 4)

# Initial values
inits <- function() list(mu = runif(1, 0, 4), beta1 = runif(1, -1, 1),
   beta2 = runif(1, -1, 1), alpha = runif(235, -2, 2), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "beta1", "beta2", "alpha", "eps", "sd.alpha",
   "sd.eps")

# MCMC settings
ni <- 12000
nt <- 6
nb <- 6000
nc <- 3

# Call WinBUGS from R (BRT 7 min)
out6 <- bugs(win.data, inits, params, "GLMM4.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out6, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mu	2.09	0.10	1.88	2.02	2.10	2.17	2.28	1.06	64
beta1	0.00	0.08	-0.15	-0.06	-0.01	0.04	0.18	1.03	210
beta2	0.00	0.02	-0.04	-0.02	0.00	0.01	0.03	1.00	1500
alpha[1]	-2.12	0.34	-2.82	-2.34	-2.10	-1.88	-1.52	1.00	610
alpha[2]	-2.36	0.37	-3.15	-2.60	-2.34	-2.11	-1.67	1.00	3000
alpha[3]	0.36	0.13	0.10	0.27	0.36	0.45	0.63	1.02	110

[. . .]

alpha[233]	1.58	0.10	1.39	1.51	1.57	1.65	1.78	1.04	67
alpha[234]	1.26	0.11	1.06	1.19	1.26	1.34	1.48	1.03	81
alpha[235]	1.49	0.10	1.30	1.42	1.49	1.57	1.70	1.03	74
eps[1]	0.00	0.10	-0.19	-0.06	0.00	0.07	0.20	1.02	140
eps[2]	0.07	0.08	-0.09	0.02	0.07	0.13	0.24	1.02	150
eps[3]	-0.16	0.07	-0.30	-0.20	-0.16	-0.11	-0.01	1.02	150
eps[4]	-0.10	0.06	-0.23	-0.14	-0.10	-0.07	0.02	1.02	170
eps[5]	0.06	0.06	-0.05	0.02	0.06	0.10	0.17	1.02	170
eps[6]	0.10	0.06	-0.03	0.07	0.10	0.14	0.21	1.01	260
eps[7]	0.22	0.07	0.07	0.18	0.23	0.27	0.35	1.01	270
eps[8]	-0.15	0.08	-0.34	-0.20	-0.15	-0.10	-0.01	1.01	270
eps[9]	-0.06	0.10	-0.29	-0.12	-0.05	0.01	0.12	1.02	290
sd.alpha	1.33	0.07	1.21	1.28	1.33	1.37	1.46	1.00	3000
sd.eps	0.17	0.06	0.09	0.13	0.16	0.19	0.30	1.01	650
deviance	11241.89	22.31	11200.00	11230.00	11240.00	11260.00	11290.00	1.00	3000

The Full Model

Finally, we fit the full model with random site effects, an overall linear time trend, random observer effects, random year effects, and a fixed first-year observer effect. Based on the estimates in the previous model, we constrain the random effects standard deviations sufficiently (i.e., specify a smaller range for the uniform prior distributions) so that WinBUGS does not easily get lost numerically. We also increase chain length.

```
# Specify model in BUGS language
sink("GLMM5.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                    # Overall intercept
beta1 ~ dnorm(0, 0.01)                 # Overall trend
beta2 ~ dnorm(0, 0.01)                 # First-year observer effect

for (j in 1:nsite){
   alpha[j] ~ dnorm(0, tau.alpha)      # Random site effects
   }
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 3)

for (i in 1:nyear){
   eps[i] ~ dnorm(0, tau.eps)          # Random year effects
   }
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 1)

for (k in 1:nobs){
   gamma[k] ~ dnorm(0, tau.gamma)      # Random observer effects
   }
tau.gamma <- 1/ (sd.gamma * sd.gamma)
sd.gamma ~ dunif(0, 1)
```

```
# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- mu + beta1 * year[i] + beta2 * first[i,j] +
   alpha[j] + gamma[newobs[i,j]] + eps[i]
        } #j
    } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), nobs = 272,
   newobs = t(newobs), first = t(first), year = ((1:9)-5) / 4)

# Initial values
inits <- function() list(mu = runif(1, 0, 4), beta1 = runif(1, -1, 1),
   beta2 = runif(1, -1, 1), alpha = runif(235, -1, 1), gamma = runif(272,
   -1, 1), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "beta1", "beta2", "alpha", "gamma", "eps",
   "sd.alpha", "sd.gamma", "sd.eps")

# MCMC settings
ni <- 12000
nt <- 6
nb <- 6000
nc <- 3

# Call WinBUGS from R (BRT 11 min)
out7 <- bugs(win.data, inits, params, "GLMM5.txt", n.chains = nc,
   n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
   bugs.dir, working.directory = getwd())
```



```
# Summarize posteriors
print(out7, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mu	2.08	0.10	1.87	2.01	2.07	2.14	2.27	1.02	570
beta1	0.02	0.09	-0.16	-0.04	0.01	0.07	0.24	1.04	3000
beta2	0.02	0.02	-0.02	0.01	0.02	0.04	0.07	1.00	3000
alpha[1]	-2.43	0.37	-3.18	-2.66	-2.41	-2.18	-1.75	1.00	790
alpha[2]	-2.10	0.41	-2.92	-2.37	-2.08	-1.81	-1.33	1.01	210
alpha[3]	0.36	0.26	-0.15	0.20	0.36	0.53	0.87	1.01	400
[...]									
alpha[233]	1.63	0.30	1.04	1.43	1.62	1.83	2.25	1.01	300
alpha[234]	1.55	0.20	1.17	1.41	1.54	1.69	1.94	1.04	60
alpha[235]	1.78	0.20	1.41	1.64	1.77	1.92	2.17	1.04	61
gamma[1]	0.06	0.25	-0.44	-0.10	0.06	0.23	0.58	1.00	3000
gamma[2]	-0.01	0.34	-0.70	-0.24	-0.02	0.22	0.65	1.00	610
gamma[3]	-0.50	0.27	-1.05	-0.68	-0.50	-0.32	0.00	1.00	1800
[...]									

gamma[270]	-0.57	0.28	-1.15	-0.75	-0.56	-0.38	-0.03	1.00	1600
gamma[271]	0.02	0.23	-0.44	-0.14	0.02	0.17	0.48	1.00	1300
gamma[272]	-0.27	0.20	-0.66	-0.41	-0.28	-0.13	0.08	1.07	36
eps[1]	0.01	0.10	-0.19	-0.05	0.01	0.07	0.24	1.03	220
eps[2]	0.10	0.09	-0.08	0.04	0.09	0.15	0.28	1.03	130
eps[3]	-0.14	0.07	-0.29	-0.18	-0.14	-0.10	0.01	1.03	75
eps[4]	-0.09	0.06	-0.22	-0.13	-0.10	-0.06	0.02	1.04	55
eps[5]	0.06	0.06	-0.06	0.02	0.06	0.09	0.16	1.05	45
eps[6]	0.09	0.06	-0.04	0.06	0.09	0.13	0.21	1.06	52
eps[7]	0.21	0.08	0.05	0.16	0.21	0.25	0.35	1.06	66
eps[8]	-0.16	0.09	-0.36	-0.22	-0.15	-0.10	0.02	1.06	110
eps[9]	-0.07	0.11	-0.30	-0.14	-0.06	0.00	0.14	1.06	130
sd.alpha	1.31	0.07	1.18	1.26	1.30	1.35	1.45	1.00	3000
sd.gamma	0.34	0.03	0.28	0.32	0.34	0.36	0.40	1.00	640
sd.eps	0.17	0.06	0.09	0.13	0.15	0.19	0.32	1.02	130
deviance	10687.47	30.28	10630.00	10670.00	10690.00	10710.00	10750.00	1.00	2200

It appears that there is much variation in the counts of coal tits due to differences among sites and also due to differences among observers. We note, though, that site and observer effects are confounded to some degree, since only one to a few observers are tested on each site and that observers are not randomly allocated to sites. There is a fairly small variance component due to years. Neither year as a continuous covariate (beta1) nor first-year observer effects (beta2) seem important, since their 95% CRI covers zero by a wide margin. Regarding the latter, we repeat that first-year observer effects are confounded with the effect of the first year. For a more thorough assessment of first-year observer effects, we might want to repeat the analysis for years 2–9 only.

This concludes our overview of different variants of a Poisson GLMM. We hasten to say that the models in this chapter are purely phenomenological and not necessarily the best models for animal counts. For instance, we have not built in any population dynamics: changes from year to year are not autocorrelated as one might expect owing to the effects of density dependence or of the vital rates inducing these changes. For models that include such additional ecological realism, see Chapters 5 and 11.

4.4 SUMMARY AND OUTLOOK

In this chapter, we have introduced another crucial concept (besides the GLM) of applied statistical models: random effects. We have seen that the inclusion of random effects changes the scope of inference in an analysis, allows one to study the variability in parameter sets, accounts for internal structure (correlations) in the data, achieves a more honest accounting for the uncertainties in a modeled system, and may result in improved estimates of each parameter owing to "borrowing strength

from the ensemble". We hope that the simulation of data sets and model specification in the BUGS language has enhanced your understanding of the actual meaning of random effects, something which we believe is easily lost when specifying random-effects models in software such as R or GenStat. The setting of all random effects in this chapter was the generalized linear model (GLM); hence, we dealt with GLMMs. Poisson GLMMs now form one of the standard models for counts of animals; see, for example, the papers by Link and Sauer (2002), Sauer and Link (2002), Ver Hoef and Jansen (2007), and Cressie et al. (2009). They are hierarchical models and to some (e.g., Cressie et al., 2009) have become almost synonymous with "the hierarchical model" in ecology. However, there are many different hierarchical models in ecology, and we will encounter a variety of hierarchical or random-effects models in almost every chapter in this book.

The Poisson GLMMs in this chapter are useful, for instance, because they more properly partition the effects of multiple sources of variability in a modeled system. One way how a partitioning of the system variability can be achieved is to separate out variability intrinsic to the studied system and variability associated with the measurement, or the observation, of that system. Hence, Poisson GLMMs achieve a certain degree of partitioning of the data into what may be called the ecological process and the observation process.

However, we believe that these models have important shortcomings as a general framework for inference about animal or plant population sizes. Most of all, they cannot directly account for that hallmark of ecological field data, namely, imperfect detection. Models such as those in this chapter do not contain an explicit description of a meaningful ecological process; they are "implicit" hierarchical models in the sense of Royle and Dorazio (2008). Their state is some sort of "expected count", and it is difficult to attach a clear biological interpretation to that. In other words, the expected count is always a result of a certain observation process and will depend to a large extent on how good a job you do at the counting. This can hardly be claimed to be of ecological interest!

A variant of the implicit hierarchical models in this chapter has come to be called "state-space models" in population dynamics and is the subject of the next chapter. They go one step further in introducing biological realism into the modeling, in that they make a better distinction between what they call process and observation models. Furthermore, their process model usually contains added biological realism in the form of an autoregressive population model and may contain a parameter for density dependence. These are powerful and interesting models, and yet, they fail to explicitly account for imperfect detection and its effects on the inference about the population processes. To make a clear and ecologically meaningful distinction between the ecological and the observation

processes, other protocols and models must be chosen, leading to what Royle and Dorazio (2008) called "explicit" hierarchical models. These models exploit explicit information about the observation process, enabling one to explicitly model the observation process by estimating detection probability. In the context of statistical inference about animal numbers, the natural generalization of an implicit Poisson GLMM is the class of binomial mixture models introduced in Chapter 12.

4.5 EXERCISES

1. Overdispersion: Generate a data set using the function in Section 4.2.1 and use WinBUGS to compare the regression estimates under the Poisson GLM and those under the Poisson GLMM. The Bayesian analysis yields better estimates of the uncertainty in the estimates of a random-effects model and lets you see more clearly how the regression estimates have an increased posterior standard deviation when estimated under the model with random year effects.
2. First-year observer effect: We have seen that in the tit data, any first-year observer effect is confounded with the effect of the first year. Repeat the last analysis for a restricted data set without year 1.
3. Reparameterizations: In GLMM 2, put the grand mean of the double random-effects model, mu, into the hyperdistribution of one of the random effects, that is, fit the model like the following:

```
for (j in 1:nsite){
alpha[j] ~ dnorm(mu, tau.alpha)
}
```

   You will see that convergence is worse. This is an example of where WinBUGS is very sensitive to how a model is parameterized.
4. Interpretation of random effects: Fit a series of models to the tit data with different random effects:
   - A site random effect: random contributions from each site
   - A year random effect: random contributions from each year
   - A site plus a year random effect
   - A site-by-year random effect: random contributions from each site–year combination

   Compare parameter estimates, explain the difference in the interpretation of those models, and try to make sense of the differences.
5. Fixed and random: Convert these models into fixed-effects models, that is, specify each of the following models without making the assumption that a set of effects come from a common distribution: site, year, site + year, observer, site + observer, year + observer,

site + year + observer, first-year indicator. Comparing the fixed- and the random-effects version of a model as specified in the BUGS language will be very helpful for your understanding of mixed models!

6. Covariates: In the tit model in Section 4.3.2, add the log-linear effects of elevation and forest cover in the linear predictor of abundance. Also add in squared effects of these covariates.
7. Take the model and the data from Section 4.3.1.
   a. Drop the quadratic and the cubic polynomial terms of year.
   b. Next, turn the random-effects Poisson GLM into a fixed- (year and site) effects Poisson GLM, that is, drop the randomness assumption for site and year. Hint: your model will then be a two-way, main-effects ANOVA, so you will have to constrain some parameters to make it identifiable.
8. Take the model and the data from Section 4.3.1 and model the response as coming from a normal distribution. This will clarify some of the differences between a normal and a Poisson GLMM.

This page intentionally left blank