CHAPTER

12

Linear Mixed-Effects Model

OUTLINE

12.1 INTRODUCTION

Mixed-effects or mixed models contain factors, or more generally covariates, with both fixed and random effects. During the last 15 years or so, the use of mixed models has greatly increased in statistical applications in ecology and related disciplines (Pinheiro and Bates, 2000; McCulloch and Searle, 2001; Lee et al., 2006; Littell et al., 2008). As explained in Chapter 9, there may be at least three benefits to assuming

Introduction to WinBUGS for Ecologists
DOI: 10.1016/B978-0-12-378605-0.00012-0

FIGURE 12.1 Gravid female asp viper (*Vipera aspis*), France, 2008. (*Photo T. Ott*)

a set of parameters constitutes a random sample from some distribution, whose hyperparameters are then estimated as the main structural parameters of a model: increased scope of the inference, more honest accounting for system uncertainty, and efficiency of estimation.

In Chapter 9, we met a one-way analysis of variance model that, apart from an overall mean, contained only random effects and could be called a variance-components model. It could also be called a mixed model if the overall mean, the intercept, is viewed as a fixed effect, but this terminology is not standard. Here, we consider a classic mixed model that arises as a direct generalization of the analysis of covariance (ANCOVA) model in Chapter 11. We modify our asp viper (Fig. 12.1) data set from there just a bit and assume we now have measurements from a much larger number of populations, say, 56. A random-effects factor need not possess that many levels (some statisticians even fit a two-level factor such as sex as random; see Gelman, 2005), but one rarely sees fewer than, say, 5–10 or so parameters fitted as random effects. Estimating a variance with so few values, which are moreover unobserved, will not result in very precise and perhaps biased estimates (see also Lambert et al., 2005).

We resimulate some asp viper data using R code fairly similar to that in the previous chapter. However, we now constrain the values for at least one set of effects (intercepts and/or slopes) to come from a normal distribution: this is what the random-effects assumption means. There are at least three sets of assumptions that one may make about the random effects for the intercept and/or the slope of regression lines that are fitted to grouped (here, population-specific) data:

1. Only intercepts are random, but slopes are identical for all groups.
2. Both intercepts and slopes are random, but they are independent.
3. Both intercepts and slopes are random, and there is a correlation between them.

(An additional case, where slopes are random and intercepts are fixed, is not a sensible model in most circumstances.) Model No. 1 is often called a random-intercepts model, and both models No. 2 and 3 are also called random-coefficients models. Model No. 3 is the default in R's function `lmer()` in package lme4 when fitting a random-coefficients model.

We now first generate a random-coefficients data set under model No. 2, where both intercepts and slopes are uncorrelated random effects. We then fit both a random-intercepts (No. 1) and a random-coefficients model without correlation (No. 2) to these data (see Sections 12.2–12.4). Then, we generate a second data set that includes a correlation between random intercepts and random slopes and adopt the random-coefficients model with correlation between intercepts and slopes (No. 3) to analyze it (see Section 12.5).

This is a key chapter for your understanding of mixed models, and I expect its contents to be helpful for the general understanding of mixed models to many ecologists. A close examination of how such data can be assembled (i.e., simulated) will be an invaluable help for understanding how analogous data sets are broken down (i.e., analyzed) using mixed models. Indeed, I believe that very few strategies can be more effective to understand this type of mixed model than the combination of simulating data sets and describing the models fitted in WinBUGS syntax.

Here is one way in which to write the random-coefficients model without correlation between the random effects for mass y_i of snake i in population j:

$$
\begin{aligned}
& y_i = \alpha_{j(i)} + \beta_{j(i)} * x_i + \varepsilon_i \\
& \alpha_j \sim \text{Normal}(\mu_\alpha, \sigma_\alpha^2) && \text{\# Random effects for intercepts} \\
& \beta_j \sim \text{Normal}(\mu_\beta, \sigma_\beta^2) && \text{\# Random effects for slopes} \\
& \varepsilon_i \sim \text{Normal}(0, \sigma^2) && \text{\# Residual ``random'' effects}
\end{aligned}
$$

Exactly as in the ANCOVA model in Chapter 11, mass y_i is related to body length x_i of snake i by a straight-line relationship with population-specific values for intercept α_j and slope β_j. (These regression parameters vary by individual i according to their membership to population j.) However, both α_j and β_j are now assumed to come from an independent normal distribution, with means μ_α and μ_β and variances of σ_α^2 and σ_β^2, respectively.

The residuals ε_i for snake i are assumed to come from another independent normal distribution with variance σ^2.

12.2 DATA GENERATION

As always, we assume a balanced design for simple convenience, though that is not required to conduct mixed model analyses using restricted maximum likelihood (REML) in R or a Bayesian analysis in WinBUGS. Indeed, the flexibility with unbalanced data sets was one of the main reasons why REML-based mixed model estimation became so much more popular than estimation based on sums of squares decompositions (Littell et al., 2008).

```
n.groups <- 56                  # Number of populations
n.sample <- 10                  # Number of vipers in each pop
n <- n.groups * n.sample              # Total number of data points
pop <- gl(n = n.groups, k = n.sample) # Indicator for population
```

We directly normalize covariate `length` to avoid trouble with WinBUGS.

```
# Body length (cm)
original.length <- runif(n, 45, 70)
mn <- mean(original.length)
sd <- sd(original.length)
cat("Mean and sd used to normalise.original length:", mn, sd, "\n\n")
length <- (original.length – mn) / sd
hist(length, col = "grey")
```

We build a design matrix without intercept.

```
Xmat <- model.matrix(~pop*length–1–length)
print(Xmat[1:21,], dig = 2)           # Print top 21 rows
```

Next, we choose parameter values, but this time, we need to constrain them, i.e., both values for the intercepts and those for the slopes will be drawn from two normal distributions for whom we specify four hyperparameters, i.e., two means (corresponding to μ_α and μ_β) and two standard deviations (SDs) (corresponding to the square root of σ_α^2 and σ_β^2). As residual variation, we use a mean-zero normal distribution with SD of 30.

```
intercept.mean <- 230         # mu_alpha
intercept.sd <- 20            # sigma_alpha
slope.mean <- 60              # mu_beta
slope.sd <- 30                # sigma_beta

intercept.effects<-rnorm(n = n.groups, mean = intercept.mean, sd = intercept.sd)
slope.effects <- rnorm(n = n.groups, mean = slope.mean, sd = slope.sd)
all.effects <- c(intercept.effects, slope.effects)   # Put them all together
```

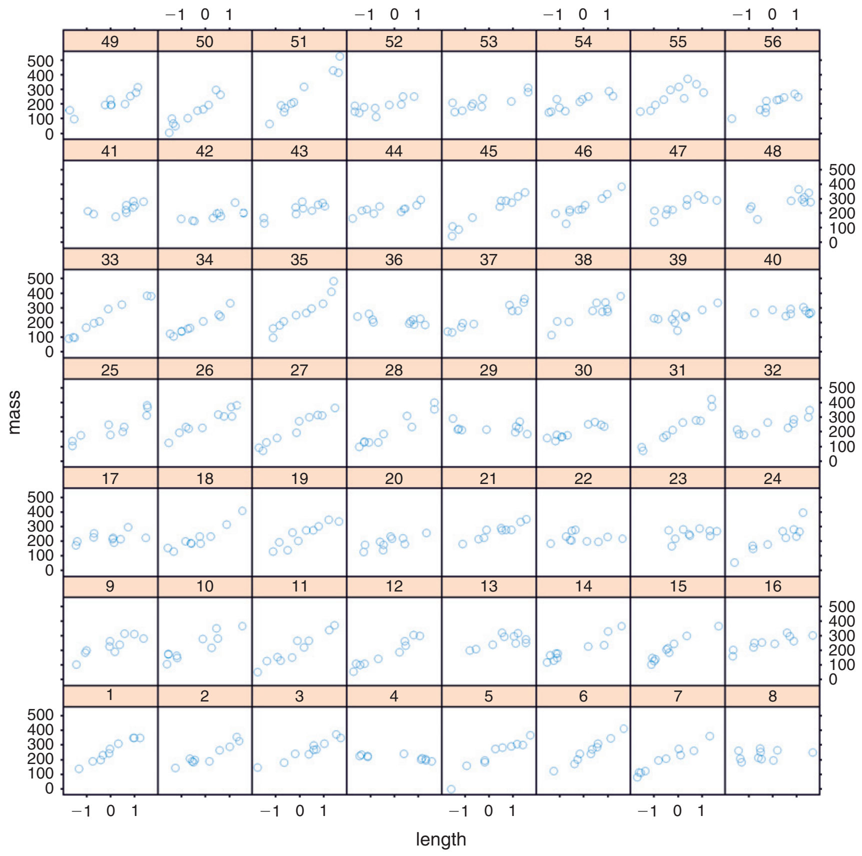

FIGURE 12.2 Trellis plot of the mass–length relationships in 56 asp viper populations (length has been standardized).

We assemble the measurements y_i as before.

```
lin.pred <- Xmat[,] %*% all.effects          # Value of lin.predictor
eps <- rnorm(n = n, mean = 0, sd = 30)       # residuals
mass <- lin.pred + eps                       # response = lin.pred + residual

hist(mass, col = "grey")                     # Inspect what we've created
```

We produce a trellis graph of the relationships in all `ngroup` populations (Fig. 12.2). Depending on the particular realization of the simulated stochastic system, we generally have quite a few fatties and may even have a few negative masses, but this doesn't really matter for our analysis.

```
library("lattice")
xyplot(mass ~ length | pop)
```

We can detect straight-line relationships between mass and length that differ among the 56 populations. What we can't see is the random-effects assumption built into the data set. That is, we are unable to distinguish a simple ANCOVA data set as in Chapter 11 from a mixed model data set as in this chapter.

12.3 ANALYSIS UNDER A RANDOM-INTERCEPTS MODEL

12.3.1 REML Analysis Using R

We first assume that the slope of the mass–length relationship is identical in all populations and that only the intercepts differ randomly from one population to another.

```
library('lme4')
lme.fit1 <- lmer(mass ~ length + (1 | pop), REML = TRUE)
lme.fit1
> lme.fit1
Linear mixed model fit by REML
Formula: mass ~ length + (1 | pop)
  AIC  BIC logLik deviance REMLdev
 5873 5890  -2932     5872    5865
Random effects:
 Groups    Name        Variance  Std.Dev.
 pop       (Intercept)  260.60    16.143
 Residual              1930.94    43.942
Number of obs: 560, groups: pop, 56

Fixed effects:
             Estimate Std. Error t value
(Intercept)   226.527      2.846   79.59
length         59.647      1.916   31.13

Correlation of Fixed Effects:
       (Intr)
length 0.000
```

12.3.2 Bayesian Analysis Using WinBUGS

In our Bayesian analysis of the random-intercepts model, we use a suitably wide uniform distribution as a prior for the standard deviation of the random-effects distribution (Gelman, 2006).

```
# Write model
sink("lme.model1.txt")
cat("
model {

# Priors
 for (i in 1:ngroups){
    alpha[i] ~ dnorm(mu.int, tau.int)      # Random intercepts
 }

 mu.int ~ dnorm(0, 0.001)     # Mean hyperparameter for random intercepts
 tau.int <- 1 / (sigma.int * sigma.int)
 sigma.int ~ dunif(0, 100)    # SD hyperparameter for random intercepts

 beta ~ dnorm(0, 0.001)       # Common slope
 tau <- 1 / ( sigma * sigma)  # Residual precision
 sigma ~ dunif(0, 100)        # Residual standard deviation

# Likelihood
 for (i in 1:n) {
    mass[i] ~ dnorm(mu[i], tau)                       # The random variable
    mu[i] <- alpha[pop[i]] + beta* length[i]        # Expectation
 }
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(mass = as.numeric(mass), pop = as.numeric(pop), length = length,
ngroups = max(as.numeric(pop)), n = n)

# Inits function
inits <- function(){list(alpha = rnorm(n.groups, 0, 2), beta = rnorm(1, 1, 1),
mu.int = rnorm(1, 0, 1), sigma.int = rlnorm(1), sigma = rlnorm(1))}

# Parameters to estimate
parameters <- c("alpha", "beta", "mu.int", "sigma.int", "sigma")

# MCMC settings
ni <- 2000
nb <- 500
nt <- 2
nc <- 3

# Start Gibbs sampling
out <- bugs(win.data, inits, parameters, "lme.model1.txt", n.thin=nt, n.chains=nc,
n.burnin=nb, n.iter=ni, debug = TRUE)
```

```
# Inspect results
print(out, dig = 3)
> print(out, dig = 3)
Inference for Bugs model at "lme.model.txt", fit using WinBUGS,
 3 chains, each with 2000 iterations (first 500 discarded), n.thin = 2
 n.sims = 2250 iterations saved
              mean    sd    2.5%     25%    50%     75%   97.5%  Rhat n.eff
[...]
beta        59.348 1.948  55.457  58.030  59.33  60.700  62.978 1.002  1300
mu.int     224.606 2.925 218.600 222.700 224.70 226.600 230.077 1.001  2200
sigma.int   16.507 2.760  11.582  14.580  16.42  18.230  22.356 1.012   180
sigma       44.120 1.409  41.570  43.150  44.08  45.040  46.958 1.002  1600
[...]
```

```
# Compare with input values
intercept.mean  ;  slope.mean  ;  intercept.sd  ;  slope.sd  ;  sd(eps)
> intercept.mean  ;  slope.mean  ;  intercept.sd  ;  slope.sd  ;  sd(eps)
[1] 230
[1] 60
[1] 20
[1] 30
[1] 29.86372
```

As usual with vague priors, the two analyses yield rather comparable results. Interestingly, the residual standard deviation in both is estimated too high. This is because we simulated the data to contain random variation among the slopes, but we did not fit this model, so this variation is unaccounted for and gets absorbed in the residual.

12.4 ANALYSIS UNDER A RANDOM-COEFFICIENTS MODEL WITHOUT CORRELATION BETWEEN INTERCEPT AND SLOPE

12.4.1 REML Analysis Using R

Next, we assume that both slopes and intercepts of the mass–length relationship differ among populations in the fashion of two independent random variables, i.e., we assume the absence of a correlation between intercept and slope. Thus, we will analyze the data under the same model that we used to generate our data set.

```
library('lme4')
lme.fit2 <- lmer(mass ~ length + (1 | pop) + ( 0+ length | pop))
lme.fit2
```

```
> lme.fit2
Linear mixed model fit by REML
```

```
Formula: mass ~ length + (1 | pop) + (0 + length | pop)
   AIC   BIC  logLik  deviance  REMLdev
  5598  5619   -2794      5596     5588
Random effects:
    Groups     Name           Variance    Std.Dev.
    pop        (Intercept)     274.37     16.564
    pop        length         1012.49     31.820
    Residual                   875.96     29.597
Number of obs: 560, groups: pop, 56

Fixed effects:
             Estimate  Std. Error  t value
(Intercept)   228.698       2.579    88.68
length         59.774       4.461    13.40

Correlation of Fixed Effects:
       (Intr)
length -0.002
```

12.4.2 Bayesian Analysis Using WinBUGS

Finally, here is the Bayesian analysis of the simple random-coefficients model:

```
# Define model
sink("lme.model2.txt")
cat("
model {

# Priors
 for (i in 1:ngroups){
      alpha[i] ~ dnorm(mu.int, tau.int)      # Random intercepts
      beta[i] ~ dnorm(mu.slope, tau.slope)    # Random slopes
 }

 mu.int ~ dnorm(0, 0.001)      # Mean hyperparameter for random intercepts
 tau.int <- 1 / (sigma.int * sigma.int)
 sigma.int ~ dunif(0, 100)     # SD hyperparameter for random intercepts

 mu.slope ~ dnorm(0, 0.001)    # Mean hyperparameter for random slopes
 tau.slope <- 1 / (sigma.slope * sigma.slope)
 sigma.slope ~ dunif(0, 100)   # SD hyperparameter for slopes

 tau <- 1 / ( sigma * sigma)     # Residual precision
 sigma ~ dunif(0, 100)           # Residual standard deviation

# Likelihood
 for (i in 1:n) {
    mass[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha[pop[i]] + beta[pop[i]]* length[i]
  }
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(mass = as.numeric(mass), pop = as.numeric(pop), length = length,
ngroups = max(as.numeric(pop)), n = n)

# Inits function
inits <- function(){ list(alpha = rnorm(n.groups, 0, 2), beta = rnorm(n.groups,
10, 2), mu.int = rnorm(1, 0, 1), sigma.int = rlnorm(1), mu.slope = rnorm(1, 0, 1),
sigma.slope = rlnorm(1), sigma = rlnorm(1))}

# Parameters to estimate
parameters <- c("alpha", "beta", "mu.int", "sigma.int", "mu.slope", "sigma.
slope", "sigma")

# MCMC settings
ni <- 2000
nb <- 500
nt <- 2
nc <- 3

# Start Gibbs sampling
out <- bugs(win.data, inits, parameters, "lme.model2.txt", n.thin=nt, n.chains=nc,
n.burnin=nb, n.iter=ni, debug = TRUE)
```

This is still a relatively simple model that converges rapidly.

```
print(out, dig = 2)
> print(out, dig = 2)
Inference for Bugs model at "lme.model2.txt", fit using WinBUGS,
 3 chains, each with 2000 iterations (first 500 discarded), n.thin = 2
 n.sims = 2250 iterations saved
              mean   sd   2.5%    25%    50%    75%   97.5%  Rhat  n.eff
[...]
mu.int      227.22  2.68  221.90  225.50  227.20  229.00  232.40  1.00  1400
sigma.int    17.05  2.29   12.96   15.41   16.94   18.52   21.80  1.00  2200
mu.slope     58.49  4.57   49.48   55.37   58.49   61.59   66.96  1.00  2200
sigma.slope  32.48  3.39   26.50   30.16   32.24   34.63   39.74  1.00  2200
sigma        29.67  1.01   27.77   28.95   29.65   30.34   31.66  1.00  2200
 [...]
>
```

```
# Compare with input values
> intercept.mean  ;  slope.mean  ;  intercept.sd  ;  slope.sd  ;  sd(eps)
[1]  230
[1]  60
[1]  20
[1]  30
[1]  29.86372
```

The two sets of numbers agree rather nicely, as do the solutions obtained by `lmer()` and the input values. I emphasize again that using simulated data and successfully recovering the input values gives one the confidence that the analysis in WinBUGS has probably been specified correctly. For more complex models, this is helpful, since it's so easy to make mistakes!

Finally, we note that the realized values of the intercept and slope random effects are estimated and are returned by typing `ranef(lme.fit2)` for the analysis in R. They are also contained in the WinBUGS output that we get by typing `print(out, dig = 2)`.

12.5 THE RANDOM-COEFFICIENTS MODEL WITH CORRELATION BETWEEN INTERCEPT AND SLOPE

12.5.1 Introduction

The random-coefficients model with correlation is a simple extension of the previous model. The mass y_i of snake i in population j is assumed to be described by the following relations:

$$y_i = \alpha_{j(i)} + \beta_{j(i)} * x_i + \varepsilon_i$$

$$(\alpha_j, \beta_j) \sim \text{MVN}(\mu, \mathbf{\Sigma}) \qquad \text{\# Bivariate normal random effects}$$

$$\mu = (\mu_\alpha, \mu_\beta) \qquad \text{\# Mean vector}$$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_\alpha^2 & \sigma_{\alpha\beta} \\ \sigma_{\alpha\beta} & \sigma_\beta^2 \end{pmatrix} \qquad \text{\# Variance–covariance matrix}$$

$$\varepsilon_i \sim \text{Normal}(0, \sigma^2) \qquad \text{\# Residual “random” effects}$$

As before, the mass y_i of snake i in population j is related to its body length x_i by a straight-line relationship with population-specific values for intercept α_j and slope β_j. But now, pairs of α_j and β_j from the same population are assumed to come from a multivariate normal distribution (actually, here, a bivariate normal) with mean vector μ and variance–covariance matrix $\mathbf{\Sigma}$. The latter contains the variances of the intercept (σ_α^2) and the slope (σ_β^2) in the diagonal and the covariance between α_j and β_j ($\sigma_{\alpha\beta}$) in the off-diagonals. As before, the residuals ε_i for snake i are assumed to come from an independent (univariate) normal distribution with variance σ^2. The interpretation of the covariance is such that positive values indicate a steeper mass–length relationship for snakes with a greater mass.

12.5.2 Data Generation

We generate data under the random-coefficients model.

```
n.groups <- 56
n.sample <- 10
n <- n.groups * n.sample
pop <- gl(n = n.groups, k = n.sample)
```

We generate the covariate `length`.

```
original.length <- runif(n, 45, 70) # Body length (cm)
mn <- mean(original.length)
sd <- sd(original.length)
cat("Mean and sd used to normalise.original length:", mn, sd, "\n\n")
length <- (original.length - mn) / sd
hist(length, col = "grey")
```

We build the same design matrix as before.

```
Xmat <- model.matrix(~pop*length-1-length)
print(Xmat[1:21,], dig = 2)          # Print top 21 rows
```

We choose the parameter values, i.e., the population-specific intercepts and slopes from a bivariate normal distribution (available in the R package MASS) whose hyperparameters (two means and the four cells of the variance–covariance matrix) we need to specify. We use again as residual variation a mean-zero normal distribution with SD of 30.

```
library(MASS)                        # Load MASS
?mvrnorm                             # Calls help file

intercept.mean <- 230                # Values for five hyperparameters
intercept.sd <- 20
slope.mean <- 60
slope.sd <- 30
intercept.slope.covariance <- 10

mu.vector <- c(intercept.mean, slope.mean)
var.cova.matrix <- matrix(c(intercept.sd^2,intercept.slope.covariance,
intercept.slope.covariance, slope.sd^2),2,2)

effects <- mvrnorm(n = n.groups, mu = mu.vector, Sigma = var.cova.matrix)
effects                          # Look at what we've created
apply(effects, 2, mean)
var(effects)
```

```
intercept.effects <- effects[,1]
slope.effects <- effects[,2]
all.effects <- c(intercept.effects, slope.effects)      # Put them all together
```

Assemble the measurements y_i.

```
lin.pred <- Xmat[,] %*% all.effects             # Value of lin.predictor
eps <- rnorm(n = n, mean = 0, sd = 30)          # residuals
mass <- lin.pred + eps                          # response = lin.pred + residual

hist(mass, col = "grey")                        # Inspect what we've created
```

Again, negative masses are possible for some realizations of the data set. We look at the simulated data set:

```
library("lattice")
xyplot(mass ~ length | pop)
```

Now, we analyze this second data set allowing for a nonzero covariance between intercept and slope effects.

12.5.3 REML Analysis Using R

The model with an intercept–slope correlation is the default when specifying a random-coefficients model in R using function `lmer()`.

```
library('lme4')
lme.fit3 <- lmer(mass ~ length + (length | pop))
lme.fit3
> lme.fit3
Linear mixed model fit by REML
Formula: mass ~ length + (length | pop)
  AIC   BIC  logLik  deviance  REMLdev
 5624  5650  -2806      5620     5612
Random effects:
 Groups    Name         Variance  Std.Dev.  Corr
 pop       (Intercept)  255.86    15.996
           length       652.01    25.534    0.333
 Residual               979.29    31.294
Number of obs: 560, groups: pop, 56

Fixed effects:
               Estimate  Std. Error  t value
(Intercept)    233.342      2.554    91.38
length          68.811      3.698    18.61

Correlation of Fixed Effects:
       (Intr)
length 0.254
```

12.5.4 Bayesian Analysis Using WinBUGS

Here is one way in which to specify a Bayesian analysis of the random-coefficients model with correlation. For a different and more general way to allow for correlation among two or more sets of random effects in a model, see Gelman and Hill (2007, p. 376–377).

```
# Define model
sink("lme.model3.txt")
cat("
model {

# Priors
 for (i in 1:ngroups){
    alpha[i] <- B[i,1]
    beta[i] <- B[i,2]
    B[i,1:2] ~ dmnorm(B.hat[i,], Tau.B[,])
    B.hat[i,1] <- mu.int
    B.hat[i,2] <- mu.slope
}

 mu.int ~ dnorm(0, 0.001)      # Hyperpriors for random intercepts
 mu.slope ~ dnorm(0, 0.001)    # Hyperpriors for random slopes

 Tau.B[1:2,1:2] <- inverse(Sigma.B[,])
 Sigma.B[1,1] <- pow(sigma.int,2)
 sigma.int ~ dunif(0, 100)     # SD of intercepts
 Sigma.B[2,2] <- pow(sigma.slope,2)
 sigma.slope ~ dunif(0, 100)   # SD of slopes
 Sigma.B[1,2] <- rho*sigma.int*sigma.slope
 Sigma.B[2,1] <- Sigma.B[1,2]
 rho ~ dunif(-1,1)
 covariance <- Sigma.B[1,2]

 tau <- 1 / ( sigma * sigma)     # Residual
 sigma ~ dunif(0, 100)           # Residual standard deviation

# Likelihood
 for (i in 1:n) {
    mass[i] ~ dnorm(mu[i], tau)                 # The "residual" random variable
    mu[i] <- alpha[pop[i]] + beta[pop[i]]* length[i]     # Expectation
 }
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(mass = as.numeric(mass), pop = as.numeric(pop), length = length,
ngroups = max(as.numeric(pop)), n = n)
```

```
# Inits function
inits <- function(){ list(mu.int = rnorm(1, 0, 1), sigma.int = rlnorm(1), mu.slope
= rnorm(1, 0, 1), sigma.slope = rlnorm(1), rho = runif(1, −1, 1), sigma =
rlnorm(1))}

# Parameters to estimate
parameters <- c("alpha", "beta", "mu.int", "sigma.int", "mu.slope", "sigma.slope",
"rho", "covariance", "sigma")

# MCMC settings
ni <- 2000
nb <- 500
nt <- 2
nc <- 3

# Start Gibbs sampler
out <- bugs(win.data, inits, parameters, "lme.model3.txt", n.thin=nt, n.chains=nc,
n.burnin=nb, n.iter=ni, debug = TRUE)
```

We inspect the results and compare them with the frequentist analysis in Section 12.5.3 and find the usual comforting agreement between the two approaches (note `rho` in the Bayesian analysis has to be compared with `Corr` in the frequentist analysis).

```
print(out, dig = 2)          # Bayesian analysis
lme.fit3                     # Frequentist analysis
```

As usual, the approximate solution given by `lmer()` comes reasonably close to the exact solution from the Bayesian analysis (Gelman and Hill, 2007). Even though convergence is achieved fairly rapidly in the latter, it often takes much longer to obtain the exact Bayesian solution in a mixed model analysis. So there is a price to pay for enjoying the advantages of the Bayesian analysis, and this price can be fairly high when using WinBUGS to fit more complex models.

For some realizations of the data set, the covariance may be estimated at a negative value, even though we've built the data with a positive covariance in the parent (statistical) population of intercepts and slopes. This is a reflection of both sampling variation and estimation error. Also look at how imprecise the estimate for the covariance is; covariances are even harder to estimate than variances. R doesn't return a standard error for that estimate.

12.6 SUMMARY

We have introduced the classic mixed ANCOVA model with random intercepts, random slopes, and the possibility of an intercept–slope covariance. Understanding the material presented in this chapter is essential for a thorough understanding of much of the current mixed modeling in ecology. The ideas presented here appear over and over again, in later chapters of this book, as well as in the applied work of many quantitative ecologists.

EXERCISES

1. *Specification of fixed- and random-effects in WinBUGS*: The WinBUGS model description for the random-intercepts, random-slope model (i.e., the second one we fit in this chapter) is very similar to the fixed-effects "version" of the same model, i.e., the one we fitted in Chapter 11. Without looking at the WinBUGS model description in that chapter, take the linear mixed model description for WinBUGS from the current chapter and change it back to a fixed-effects ANCOVA with population-specific intercepts and slopes, i.e., corresponding to what you would fit in R as `lm(mass ~ pop*length)`.
2. *Swiss hare data*: Fit a random-coefficients regression without intercept–slope correlation to mean density (i.e., `~ population * year`, with year continuous).