CHAPTER 3

LINEAR MODELS, GENERALIZED LINEAR MODELS (GLMs), AND RANDOM EFFECTS MODELS: THE COMPONENTS OF HIERARCHICAL MODELS

CHAPTER OUTLINE

3.1 INTRODUCTION

In Chapter 2, we introduced hierarchical models (HMs) as consisting of a linked sequence of probability models for observed and unobserved random variables. We saw that the former are observed data, and the latter latent or partially latent variables or random effects such as the occurrence or abundance state of a local population, but that a random effect may also be some abstract "group effect." Typically, we model the structure in each random variable using link functions and linear functions of covariates—i.e., as simple *weighted sums of covariate values* (where coefficients represent the weights), exactly as in generalized linear models (GLMs; McCullagh and Nelder, 1989). Therefore, an HM can be described as a compound GLM—a sequence of two or more related GLMs. In this book, we use the R package `unmarked` (Fiske and Chandler, 2011) and BUGS software (Lunn et al., 2000; Plummer, 2003) for almost all fitting of HMs. Much of the power of HMs derives from the flexible ways in which we specify linear models at each level in the HM—e.g., for occupancy and for detection in an occupancy model. Thus, it is essential that you understand very well the specification of a wide range of linear models, in order to be an effective hierarchical modeler.

The main focus of this chapter is the linear model. We show how different linear models can be described in algebra and in the model definition language in R—e.g., using the functions `lm` and `glm`, which we assume you are familiar with. When fitting HMs with `unmarked`, you will specify linear models in exactly the same way regardless of the type of model. We also show how to specify the same linear models in the BUGS programming language. In so doing, we get a little ahead of ourselves because we don't formally introduce BUGS until Chapter 5. However, seeing the algebraic model description and the R and BUGS model descriptions alongside is illuminating, although you may perhaps fully understand the BUGS model specification only after you have read Chapter 5. Ecologists can find an excellent introduction to linear models in Chapter 6 of Evan Cooch's gentle introduction to program MARK (www.phidot.org/software/mark/docs/book).

We also provide a brief and applied introduction of two further crucial concepts in applied statistics that are almost always found in HMs: GLMs and random effects. Strictly speaking, random effects are nothing new in HMs: they are simply the outcome of an *unobserved* random variable, as opposed to the data that are the outcome of an *observed* random variable. The presence of an unobserved random variable is one defining feature of an HM or a mixed model. However, in our experience, ecologists often find random-effects or mixed models confusing, and hence we attempt to shed some light on this class of models here. We focus on GLMs and random effects mostly as they are relevant for your understanding of the types of HMs in this book. Our coverage of both topics is more summary than, for instance, in Chapters 3 and 4 of Kéry and Schaub (2012) or in textbooks such as McCullagh and Nelder (1989), Dobson and Barnett (2008) or Ntzoufras (2009) for GLMs, or in books like Pinheiro and Bates (2000), McCulloch and Searle (2001), Lee et al. (2006), Gelman and Hill (2007) or Littell et al. (2008), and in Bolker et al. (2009) for random-effects/mixed models.

Most ecologists learn statistical modeling best by example; hence, we use for illustration a toy data set and imagine that we had measured wingspan and body length of nine blue-eyed hooktail dragonflies (*Onychogomphus uncatus*; Figure 3.1) in three populations in the Spanish Pyrenees (Navarra, Aragon, and Catalonia). For each individual we also assessed sex, color intensity (proportion of body that is yellow as opposed to black), ectoparasite load (number of mites counted), and whether each of the four wings (two hind, two front) was damaged. Part of this toy data set and analysis is taken from Chapters 3 and 4 in Kéry and Schaub (2012).

In Section 3.2, we illustrate linear models by investigating the relationships between wingspan (a numerical metric response) and four explanatory variables: population, sex, body length, and color intensity. The first two are factors with three (population: 1 = Navarra, 2 = Aragon, 3 = Catalonia) and two (sex: 1 = male; 2 = female) levels, respectively. Factors are categorical explanatory variables in which numbers have no quantitative meaning; they are merely labels or group names. In contrast,

FIGURE 3.1

Male blue-eyed hooktail (*Onychogomphus uncatus*), Aragon/Spain, 2013. *(Photo: M. Kéry.)*

body length and color intensity are continuous explanatory variables, or covariates, where numbers do have a quantitative meaning. For instance, we can compute a difference in body length between two dragonflies, but we cannot compute a difference between their sexes. In this chapter, we will distinguish continuous and categorical explanatory variables by use of the terms "*covariates*" and "*factors*." We show how their pairwise effects are combined in linear models in an additive or interactive fashion. Of course, in your modeling, you will typically have more than just two explanatory variables, but the principles are best explained in the simplest possible setting.

In Section 3.3, we will use parasite load and wing damage as response variables to illustrate GLMs. Finally, in Section 3.4, we revisit some models from Sections 3.2 and 3.3 and turn them into random-effects, hierarchical, or generalized linear mixed models (GLMMs).

```
# Define data
pop <- factor(c(rep("Navarra", 3), rep("Aragon", 3), rep("Catalonia", 3)), levels
= c("Navarra", "Aragon", "Catalonia"))                          # Population
wing <- c(10.5, 10.6, 11.0, 12.1, 11.7, 13.5, 11.4, 13.0, 12.9) # Wingspan
```

```
body <- c(6.8, 8.3, 9.2, 6.9, 7.7, 8.9, 6.9, 8.2, 9.2)  # Body length
sex <- factor(c("M","F","M","F","M","F","M","F","M"), levels = c("M", "F"))
mites <- c(0, 3, 2, 1, 0, 7, 0, 9, 6)                   # Number of ectoparasites
color <- c(0.45, 0.47, 0.54, 0.42, 0.54, 0.46, 0.49, 0.42, 0.57) # Color intensity
damage <- c(0,2,0,0,4,2,1,0,1)                         # Number of wings damaged

cbind(pop, sex, wing, body, mites, color, damage)      # Print out data set
      pop sex wing body mites color damage
 [1,]   1   1 10.5  6.8     0  0.45      0
 [2,]   1   2 10.6  8.3     3  0.47      2
 [3,]   1   1 11.0  9.2     2  0.54      0
 [4,]   2   2 12.1  6.9     1  0.42      0
 [5,]   2   1 11.7  7.7     0  0.54      4
 [6,]   2   2 13.5  8.9     7  0.46      2
 [7,]   3   1 11.4  6.9     0  0.49      1
 [8,]   3   2 13.0  8.2     9  0.42      0
 [9,]   3   1 12.9  9.2     6  0.57      1
```

Internally in R, factor levels for `pop` and `sex` are expressed as integers ranging from 1 to the number of levels, but we can clearly see that R interprets these numbers as factor levels (i.e., mere names):

```
str(pop)
 Factor w/ 3 levels "Navarra","Aragon",..: 1 1 1 2 2 2 3 3 3
```

We plot the data for wingspan, parasite load, and damage (Figure 3.2).

```
par(mfrow = c(1, 3), cex = 1.2)
colorM <- c("red", "red", "blue", "green", "green")  # Pop color code males
colorF <- c("red", "blue", "blue", "green", "green") # Pop color code females
```

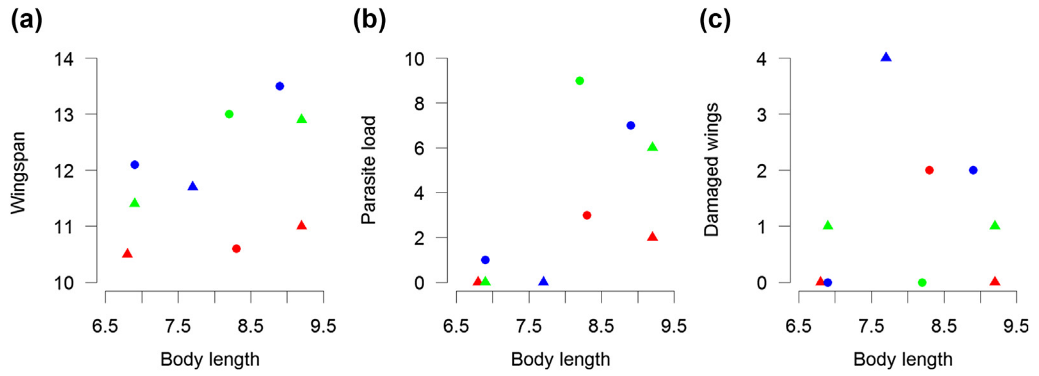

FIGURE 3.2

Relationships between wingspan (a), parasite load (b), and number of damaged wings (out of four; c), respectively, and body length, sex, and population in the dragonfly toy data set. Colors code for the three populations (Navarra—red, Aragon—green, and Catalonia—blue); circles denote females and triangles males.

```
plot(body[sex == "M"], wing[sex == "M"], col =colorM, xlim = c(6.5, 9.5), ylim = c(10, 14),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "Wingspan")
points(body[sex == "F"], wing[sex == "F"], col =colorF, pch = 16)
text(6.8, 13.8, "A", cex = 1.5)
plot(body[sex == "M"], mites[sex == "M"], col = colorM, xlim = c(6.5, 9.5), ylim = c(0, 10),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "Parasite load")
points(body[sex == "F"], mites[sex == "F"], col = colorF, pch = 16)
text(6.8, 9.7, "B", cex = 1.5)
plot(body[sex == "M"], damage[sex == "M"], col = colorM, xlim = c(6.5, 9.5), ylim = c(0, 4),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "Damaged wings")
points(body[sex == "F"], damage[sex == "F"], col = colorF, pch = 16)
text(6.8, 3.9, "C", cex = 1.5)
```

3.2 LINEAR MODELS

In a linear model, the effects of covariates or factors act on a response in a purely additive fashion—i.e., they are expressed as simple weighted sums of the values of the covariates, with the weights being the coefficients or parameters of the linear model. The model is said to be linear in the parameters, but it does not necessarily represent a straight line when shown as a graph. For instance, models with polynomial terms, such as quadratic or cubic, do not translate into a straight-line graph, but they are linear models. A linear model can be described in a variety of ways:

1. in words; e.g., when we say something like "population and body length act additively on wingspan,"
2. using specific names or labels for least-squares techniques that *imply* a certain linear model; e.g., when we say "t-test," "linear regression," "ANOVA," or "ANCOVA" (Mead, 1988),
3. in graphs; e.g., using line and bar plots,
4. in algebra; e.g., $y_i \sim Normal(\alpha_{j(i)} + \beta * x_i, \sigma^2)$,
5. using matrix algebra; e.g., $\mathbf{y} = \mathbf{X}\beta + \varepsilon$,
6. as a system of equations,
7. in the R model definition language; e.g., `lm(wing ~ pop + length)`, and finally
8. using the BUGS model definition language;

e.g.,
```
y[i] ~ dnorm(mu[i], tau)

mu[i] <- alpha[pop[i]] + beta * length[i].
```

In statistical modeling, most of us have become accustomed to defining certain types of linear models using point-and-click techniques available in many statistical packages, or a slick model definition language such as the one in the R functions `lm` and `glm`, or in all functions in the package `unmarked`. Model definition languages make your life easier, because they allow you to define a large range of linear models quickly and error-free. The big drawback is that it is easy to fit linear models without actually understanding them or knowing what their parameters mean. In addition, you may not be able to fit certain nonstandard linear models, or you may be able to do so but only in a very complicated way. In contrast, when fitting models in BUGS, no such shortcuts to defining linear models are available. Hence, you must know exactly what kind of linear model you want to fit and in what parameterization (see below). This is one of the main stumbling blocks for beginners in BUGS. On the other hand, it is a great advantage, because it forces upon you a much clearer understanding of what these linear models mean.

To illustrate, we will look at how to describe a certain linear model in all the different ways listed above, and then extend that example to summarize some typical linear models that can be fitted for covariates and factors. We will also see how to combine factors and continuous covariates in additive and interactive ways (see also Chapter 6 in Kéry, 2010). We focus on the algebraic and R language descriptions of a linear model, because you are likely to know and use R already, and this will be the way in which you specify linear models within HMs in `unmarked`. In addition, we show the syntax for fitting the same linear models in the BUGS language.

3.2.1 LINEAR MODELS WITH MAIN EFFECTS OF ONE FACTOR AND ONE CONTINUOUS COVARIATE

We use our dragonfly example to illustrate the description and fitting of a linear model where `pop` and `length` act in an additive fashion on wingspan, namely according to the linear model underlying a least-squares technique known as a "main-effects analysis of covariance (ANCOVA)." In Section 3.2.2, we extend the main-effects model to an interaction-effects ANCOVA model. We can fit the main-effects model in R by issuing the following commands:

```
summary(fm1 <- lm(wing ~ pop + body))

Coefficients:
               Estimate Std. Error t value  Pr(>|t|)
(Intercept)      6.5296     1.6437   3.973  0.01061 *
popAragon        1.8706     0.4521   4.138  0.00902 **
popCatalonia     1.7333     0.4489   3.861  0.01187 *
body             0.5149     0.1991   2.586  0.04908 *
[ ... ]
Residual standard error: 0.5498 on 5 degrees of freedom
Multiple R-squared: 0.8416,  Adjusted R-squared: 0.7465
F-statistic: 8.854 on 3 and 5 DF, p-value: 0.01916
```

How many parameters does our model have? Where are the estimates of these parameters presented in the R output? What do these parameters mean? To understand this, let's look at the description of this model in algebra. We can write this model as follows:

$$wing_i = \mu + \alpha_j + \beta * body_i + \varepsilon_i$$

$$\varepsilon_i \sim Normal\left(0, \sigma^2\right)$$

Here, $wing_i$ is the response of unit i (= wingspan of dragonfly i, or a single data point), and $body_i$ is the value of covariate `body length` for dragonfly i. The intercept μ is a constant shared by the response of all nine dragonflies. Vector α_j has three elements corresponding to the number of levels, or populations, in the factor `pop`, and contains the *effects* of each population. (We could write $\alpha_{j(i)}$ to emphasize that population membership j is a function of the individual i.) For population 1 (Navarra), we have α_1; for population 2 (Aragon), α_2; and for population 3 (Catalonia), α_3. Parameter β is the expected change in wingspan for a one-unit change in body length. The unexplained part in the wingspan of dragonfly i, which is not accounted for by population membership or body length of dragonfly i, is the residual ε_i. The residuals for all nine dragonflies are assumed to be independent draws from the same normal distribution with constant variance. The residual variance (σ^2) is also an estimated model parameter. In the R output (under `Residual standard error`), we see the square root of that variance estimate (σ).

Hence, our model has four parameters (α_j and β) to describe the expected response and one parameter (σ) for the variability of the response around that mean. Importantly, when fitting linear models, statistical packages like R internally do not differentiate between the two types of explanatory variables, covariates and factors, although factors must be declared as such. All linear models are internally represented as a multiple linear regression, where the effects of a factor are broken apart into those of two or more *indicator or dummy variables* that contain 0s and 1s only. In our case for `pop`, we have three dummy variables, one for each level (population) of the `pop` factor, to code for the presence or absence of an effect of that population in the expected response of each dragonfly.

However, the model above is overparameterized or parameter-redundant (Mead, 1988; Ntzoufras, 2009): we try to estimate one parameter too many. Specifically, we cannot estimate both the intercept μ and a full parameter vector $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\}$ of length 3. One of the population effects α_j has to be set to zero to make the model identifiable. Customarily in R, the parameter corresponding to the first factor level is set to zero—i.e., the constraint $\alpha_1 = 0$ is imposed. As a consequence, the intercept μ becomes the intercept of the regression line of dragonflies in population 1; this is the expected wingspan of a (hypothetical) dragonfly of length 0 in Navarra. The remaining two elements of the parameter vector $\boldsymbol{\alpha}$, which are not fixed but estimated, then become the differences between the intercepts of the regression lines in populations 2 and 3 relative to the intercept in population 1 (this is why seemingly only coefficient estimates for Aragon and Catalonia are shown in the R output above). In this model parameterization, population 1 (Navarra) serves as a baseline or reference, and the two parameters estimated for the population factor represent comparisons of the other populations with that one. This parameterization of the effects of the factor `pop` may therefore be called the *treatment contra*st or *effects parameterization*.

How do we write this model in the BUGS language? We will see in much more detail later (starting in Chapter 5) that BUGS is not a vectorized language: we cannot specify a linear model simply as `y ~ x` as in R. Rather, we have to specify the stochastic relationship between y and x for each element/observation in the two vectors, by looping over them from the first until the last element. Elements of vectors are indexed by square brackets, and hence our BUGS code will be this (this is *not* executable as is; and the text after the hash mark # is an explanatory comment, not part of the model specification):

```
for(i in 1:9){                                         # Loop over each individual
   wing[i] ~ dnorm(mean[i], tau)                       # Stochastic relationship
   mean[i] <- mu + alpha[pop[i]] + beta * body[i]      # Deterministic relationship
}
alpha[1] <- 0                                          # Fix effect of 1st level at 0
```

This says that the wingspan of dragonfly *i* is a draw from a normal distribution with an expected value (or mean) called `mean[i]` and precision `tau`. In BUGS, the dispersion parameter of the normal distribution is not the variance or the standard deviation, but the precision (the reciprocal of the variance). The expected wingspan of dragonfly *i*, `mean[i]`, is the sum of an intercept `mu`, a population-specific effect `alpha[pop[i]]`, and an effect `beta` of body length. Thus, inside of the loop we write exactly what we might write in algebra to describe that model (see also below for different algebraic ways of writing the same model, especially variant 2).

We make the following three remarks:

1. In a statistical model, the observed data are assumed to be the outcome of a stochastic process (see Chapter 2). In line with this, observed data in BUGS must always be stochastic quantities—i.e., stand on the left side of the twiddle or tilde symbol, indicating a stochastic relationship. Residuals are only defined implicitly, and although correct in algebra, it would *not* be correct in BUGS to write the following:

```
for(i in 1:9){
   wing[i] <- mu + alpha[pop[i]] + beta * body[i] + eps[i]
   eps[i] ~ dnorm(0, tau)
   # In BUGS this is WRONG !
}
```

   The reason is that you can't simply calculate an observed quantity such as `wing[i]`; it has to be stochastic—i.e. a draw from some distribution (though see the function `dsum` in the JAGS dialect of the BUGS language).
2. To specify effects of factor levels, or more generally parameters that are grouped in a vector or higher-dimensional array, we use *nested indexing* in the BUGS language: `alpha[pop[i]]`, where `pop` is the indexing variable. Nested indexing can be best understood when read from the inside out: for instance, dragonfly $i = 1$ has (internally in R) a value of pop[1] = 1, hence its expected (mean) wingspan will get a contribution of `alpha[1]`. As another example, dragonflies 6 and 7 have (internal) values of the `pop` factor of 2 and 3, respectively. Hence, their expected wingspans will get contributions of `alpha[2]` and `alpha[3]`, respectively. Be careful with the indices when writing a model in algebra, and similarly in the BUGS language; indexing of arrays is one of the more complicated things for a beginner to grasp. As said above, we might have written $\alpha_{j(i)}$ rather than simply α_j to clarify the algebraic description of the model. This would emphasize that the information given by index *j* (population membership of a dragonfly) is equivalent to that given by the factor `pop` in the analysis.
3. Factor levels in BUGS cannot be letters or words such as in our example here, but they must be integers ranging from 1 to the number of distinct levels. Thus, in a BUGS analysis, we would have to convert our factors `pop` and `sex` into numerical values and only (1,2,3) and (1,2), respectively, will be correct, while for instance, (1,2,4) for `pop` or (0,1) for `sex` will result in errors when using nested indexing.

There are several ways to write what is essentially the same linear model, and these are called *parameterizations* of a model; for instance, here is such a variant of the ANCOVA linear model:

$$wing_i = \alpha_j + \beta * body_i + \varepsilon_i$$

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

In this parameterization of the model, everything is the same as before, except that the model no longer has the parameter μ, and the meaning of the parameters α_j representing the effects of the levels of factor `pop` has changed. Now, all three elements of this parameter vector are estimable, since there is no longer a separate intercept (μ). They now represent directly the intercepts of the three regression lines—i.e., the expected wingspans of a dragonfly at body length 0 in the three populations. We call this parameterization of the factor `pop` the *means parameterization*, because each parameter in the factor corresponds directly to a 'group mean'. We can fit it in R by "subtracting the intercept" in the model formula of the function call:

```
summary(fm2 <- lm(wing ~ pop-1 + body))

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
popNavarra      6.5296     1.6437   3.973  0.01061 *
popAragon       8.4003     1.5916   5.278  0.00325 **
popCatalonia    8.2630     1.6437   5.027  0.00401 **
body            0.5149     0.1991   2.586  0.04908 *
[ ... ]
Residual standard error: 0.5498 on 5 degrees of freedom
Multiple R-squared: 0.9988,  Adjusted R-squared: 0.9979
F-statistic: 1053 on 4 and 5 DF, p-value: 1.694e-07
```

This is an exactly equivalent model to that from before: the parameter estimates are either the same (for `popNavarra`, `body`, and the residual standard error) or they can be obtained by adding two of the parameters in the former parameterization. Hence, the value of `popAragon` here is obtained by adding the value of the intercept and of `popAragon` in the previous parameterization (i.e., $8.4002 = 6.5296 + 1.8706$), and similar for `popCatalonia` (i.e., $8.2629 = 6.5296 + 1.7333$; differences are due to rounding). The advantage of this parameterization is perhaps ease of interpretation. However, it cannot be adopted for models with multiple factors, for which we must choose the effects parameterization or impose sum-to-zero constraints (see below and Ntzoufras, 2009).

In BUGS, the means parameterization of the model is specified simply like this:

```
for(i in 1:9){
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- alpha[pop[i]] + beta * body[i]
}
```

Hence, the BUGS code inside of the loop is again very similar to how we write the model in algebra. Indeed, it is identical to version 2 of the following three algebraically synonymous descriptions of the ANCOVA linear model

1. $wing_i \sim Normal(\alpha_j + \beta * body_i, \sigma^2)$
2. $wing_i \sim Normal(\mu_i, \sigma^2)$, with $\mu_i = \alpha_j + \beta * body_i$
3. $wing_i = \alpha_j + \beta * body_i + \varepsilon_i$, with $\varepsilon_i \sim Normal(0, \sigma^2)$

It is very important that you know how to write your linear models in algebra, because this greatly helps you understand what the parameters mean and, because the BUGS language model description very much resembles the model's representation in algebra; so much so, indeed, that once you know how to write the model in algebra, you have almost written it in the BUGS language! Furthermore, BUGS software can be sensitive to the parameterization chosen: sometimes one may work well and another not as well. Therefore, it is good to be able to switch between different model parameterizations and try which works (best).

Another parameterization of a linear model for factors imposes a sum-to-zero constraint—i.e., the effects of the factor levels are estimated such that their combined effect is zero. The effects can then be interpreted as deviations from the common mean; this again allows a separate intercept to be estimated, as for the effects parameterization. This parameterization may have some advantages for ease of interpretation in models with multiple factors and is easily coded in BUGS (see Ntzoufras, 2009).

The intercept for each population is the expected wingspan of a dragonfly of body length 0. This is of course nonsensical, and a more meaningful model is obtained when we regress wingspan not on body length, but on *centered* body length—i.e., on a transformation of original body length obtained by subtracting the sample mean of body length. The intercept is then the expected wingspan of a dragonfly at the average observed body length; a much more meaningful parameter. Moreover, centering covariates is often helpful to obtain convergence, both in maximum likelihood and in Bayesian Markov chain Monte Carlo (MCMC) analyses. Often to avoid numerical problems in `unmarked` or BUGS, we have to standardize covariates; i.e., center them and then divide the result by some amount, typically the covariate sample standard deviation. Note that while centering only affects the intercept, standardizing affects both the intercept and the slope of a covariate. When scaling a covariate with the standard deviation of the observed values of a covariate, the slope will be the expected change in the response variable for a one-unit change in the scaled covariate, which corresponds to a one-standard-deviation change of the original covariate.

After fitting a model, we may often want to make predictions (to draw figures or even understand what our model is telling us about the processes underlying our data); i.e., we may want to compute the expected response given the parameter estimates for certain values of the covariates. This can easily be done by hand, or you can use the R function `predict` to do this for you automatically, including an assessment of prediction uncertainty (standard errors). It is trivial as well in BUGS, but there you have to do this "by hand." We will see many examples of predictions with R and BUGS in this book (this is emphasized, for example, in Chapter 5), but here as a mini example we simply point out that the expected wingspan of a dragonfly of body length 8 in Aragon can be estimated using the output of `fm2` as $8.4003 + 0.5149 * 8 = 12.5195$.

In statistics books, you are likely to see linear models described in matrix algebra; for instance, as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$. What does this mean? Matrices and vectors are simply a manner in which numbers can be stored in an orderly way in an array. Hence, for our dragonfly example, $\mathbf{y}$ is a vector of length nine containing the responses—i.e., the wingspan of each dragonfly—$\mathbf{X}$ is the *design matrix* of the model (also called *model matrix* or *X matrix*), $\boldsymbol{\beta}$ is the parameter vector, and $\boldsymbol{\varepsilon}$ is the vector of residuals. We call *linear predictor* the result of the matrix multiplication of the design matrix and the parameter vector, $\mathbf{X}\boldsymbol{\beta}$; this is the expected wingspan given population membership and body length of each dragonfly, in the absence of the random noise $\boldsymbol{\varepsilon}$.

To better understand this, it is useful to write it out with the numbers of our data set plugged into the various matrices and vectors. We do this first for the effects parameterization and then for the means parameterization of our ANCOVA linear model. We will see that the structure of the linear model is exactly defined by the design matrix **X**, and that this moreover determines the interpretation of the fitted parameters. For the effects parameterization, we have for $\mathbf{y} = \mathbf{X\beta} + \boldsymbol{\varepsilon}$:

$$\begin{pmatrix} 10.5 \\ 10.6 \\ 11.0 \\ 12.1 \\ 11.7 \\ 13.5 \\ 11.4 \\ 13.0 \\ 12.9 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 6.8 \\ 1 & 0 & 0 & 8.3 \\ 1 & 0 & 0 & 9.2 \\ 1 & 1 & 0 & 6.9 \\ 1 & 1 & 0 & 7.7 \\ 1 & 1 & 0 & 8.9 \\ 1 & 0 & 1 & 6.9 \\ 1 & 0 & 1 & 8.2 \\ 1 & 0 & 1 & 9.2 \end{pmatrix} \begin{pmatrix} \mu \\ \alpha_2 \\ \alpha_3 \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \\ \varepsilon_9 \end{pmatrix}, \text{ with } \varepsilon_i \sim Normal\left(0, \sigma^2\right)$$

From left to right, we have response vector, design matrix, parameter vector, and residual vector. To be able to do matrix multiplication, the number of columns in the design matrix must always match the length of the parameter vector. The design matrix contains indicator variables (0s and 1s) for the effects of factors, and the measured covariate values for continuous or numerical explanatory variables. Here, the first column of the design matrix defines the intercept (which is equal to the expected wingspan of an individual with body length zero in population Navarra) by a vector of all 1s, while in the next two columns, the 1s indicate individuals in populations Aragon and Catalonia, respectively. In this description of the linear model, we recognize particularly clearly the structure of a multiple linear regression with as many single degree-of-freedom terms (scalar parameters) as columns in the design matrix and rows in the parameter vector.

The design matrix multiplied with the parameter vector produces another vector, η_i, called the *linear predictor*, which is the expected wingspan for each dragonfly. For dragonfly 1, it is given by $1 * \mu + 0 * \alpha_2 + 0 * \alpha_3 + 6.8 * \beta$, and hence its expected wingspan contains one contribution from μ and a contribution of 6.8 times β. Likewise, for dragonfly 9, the value of the linear predictor is $1 * \mu + 0 * \alpha_2 + 1 * \alpha_3 + 9.2 * \beta$. With a fitted model such as `fm1`, we can obtain the expected wingspan "by hand" (using the powerful `model.matrix` function, which we describe in more detail below) or by using the `predict` function after fitting the linear model using `lm`; note that `%*%` denotes matrix multiplication:

```
cbind(model.matrix(~pop+body) %*% fm1$coef, predict(fm1)) # Compare two solutions
      [,1]     [,2] # 1st is 'by hand', 2nd from function predict
1 10.03068 10.03068
2 10.80297 10.80297
3 11.26635 11.26635
4 11.95280 11.95280
5 12.36469 12.36469
6 12.98252 12.98252
7 11.81550 11.81550
8 12.48482 12.48482
9 12.99968 12.99968
```

Here is the means parameterization in the vector-matrix description:

$$\begin{pmatrix} 10.5 \\ 10.6 \\ 11.0 \\ 12.1 \\ 11.7 \\ 13.5 \\ 11.4 \\ 13.0 \\ 12.0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 6.8 \\ 1 & 0 & 0 & 8.3 \\ 1 & 0 & 0 & 9.2 \\ 0 & 1 & 0 & 6.9 \\ 0 & 1 & 0 & 7.7 \\ 0 & 1 & 0 & 8.9 \\ 0 & 0 & 1 & 6.9 \\ 0 & 0 & 1 & 8.2 \\ 0 & 0 & 1 & 9.2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \\ \varepsilon_9 \end{pmatrix}, \text{ with } \varepsilon_i \sim Normal(0, \sigma^2)$$

The expected wingspan of dragonfly 1 is given by $1 * \alpha_1 + 0 * \alpha_2 + 0 * \alpha_3 + 6.8 * \beta$, which is the same as before except for the small semantic change that we now call the intercept in population Navarra α_1 instead of μ. In contrast, for dragonfly 9 we get an expected wingspan of $0 * \alpha_1 + 0 * \alpha_2 + 1 * \alpha_3 + 9.2 * \beta$, and so we see that the intercept of the regression line in population 3 is now given by α_3, rather than by $\mu + \alpha_3$ as in the effects parameterization of the model. When we fit a linear model, we implicitly solve a system of equations subject to some constraints on the noise terms ε_i. Here is that system of equations:

$$\begin{aligned}
10.5 &= 1 * \alpha_1 + 0 * \alpha_2 + 0 * \alpha_3 + 6.8 * \beta + \varepsilon_1 \\
10.6 &= 1 * \alpha_1 + 0 * \alpha_2 + 0 * \alpha_3 + 8.3 * \beta + \varepsilon_2 \\
11.0 &= 1 * \alpha_1 + 0 * \alpha_2 + 0 * \alpha_3 + 9.2 * \beta + \varepsilon_3 \\
12.1 &= 0 * \alpha_1 + 1 * \alpha_2 + 0 * \alpha_3 + 6.9 * \beta + \varepsilon_4 \\
11.7 &= 0 * \alpha_1 + 1 * \alpha_2 + 0 * \alpha_3 + 7.7 * \beta + \varepsilon_5 \\
13.5 &= 0 * \alpha_1 + 1 * \alpha_2 + 0 * \alpha_3 + 8.9 * \beta + \varepsilon_6 \\
11.4 &= 0 * \alpha_1 + 0 * \alpha_2 + 1 * \alpha_3 + 6.9 * \beta + \varepsilon_7 \\
13.0 &= 0 * \alpha_1 + 0 * \alpha_2 + 1 * \alpha_3 + 8.2 * \beta + \varepsilon_8 \\
12.0 &= 0 * \alpha_1 + 0 * \alpha_2 + 1 * \alpha_3 + 9.2 * \beta + \varepsilon_9
\end{aligned}$$

Indeed, the least-squares model-fitting criterion used by the R function `lm` chooses the parameter values such that the sum of the squared residuals is minimized (hence the name). This technique does not make any distributional assumptions about the residuals, but the parameter estimates and standard errors are identical to those obtained using the maximum likelihood method when we make the explicit assumption that the residuals have a normal distribution.

The design matrix lies at the heart of a linear model, and it is imperative that we obtain a clear understanding of it when fitting linear models. R has the powerful `model.matrix` function, which shows you the design matrix for any linear model specified in the model definition language of R. This can be very useful to understand a model, and can even be useful when fitting the model in BUGS, for instance by clarifying what a linear model looks like (see also Sections 6.11 and 10.6 where we directly fit a design matrix in BUGS). Let's look at the design matrices of the two parameterizations of the main-effects ANCOVA model.

```
model.matrix(~ pop + body) # Effects parameterization
  (Intercept) popAragon popCatalonia body
1           1         0            0  6.8
2           1         0            0  8.3
3           1         0            0  9.2
4           1         1            0  6.9
5           1         1            0  7.7
6           1         1            0  8.9
7           1         0            1  6.9
8           1         0            1  8.2
9           1         0            1  9.2

model.matrix(~ pop-1 + body) # Means parameterization
  popNavarra popAragon popCatalonia body
1          1         0            0  6.8
2          1         0            0  8.3
3          1         0            0  9.2
4          0         1            0  6.9
5          0         1            0  7.7
6          0         1            0  8.9
7          0         0            1  6.9
8          0         0            1  8.2
9          0         0            1  9.2
```

Finally, the geometrical representation of the main-effects (i.e., additive) ANCOVA model in this section is that of a bundle of parallel regression lines, one for each population (Figure 3.3(a)). They are parallel because the slope is shared among all three populations.

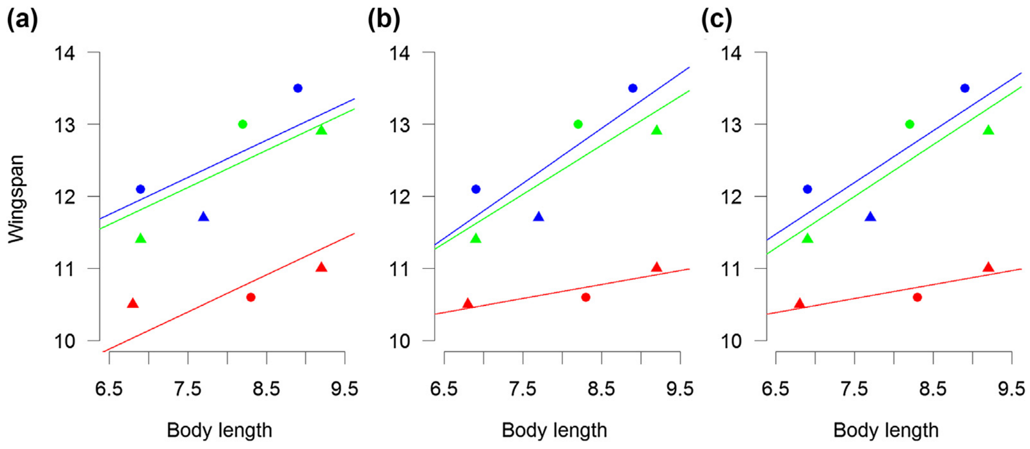

FIGURE 3.3

Geometric representation of the main-effects ANCOVA model (additive effects of population and body length; (a), the interaction-effects ANCOVA model (multiplicative effects of population and body length; b), and the partially interactive-effects ANCOVA model (c) in the *O. uncatus* toy data set. Color code and symbols as in Figure 3.2.

```
par(mfrow = c(1, 3), mar = c(5,4,2,2), cex = 1.2, cex.main = 1) # Figure 3.3 (a)
plot(body[sex == "M"], wing[sex == "M"], col = colorM, xlim = c(6.5, 9.5), ylim = c(10, 14),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "Wingspan")
points(body[sex == "F"], wing[sex == "F"], col = colorF, pch = 16)
abline(coef(fm2)[1], coef(fm2)[4], col = "red", lwd = 2)
abline(coef(fm2)[2], coef(fm2)[4], col = "blue", lwd = 2)
abline(coef(fm2)[3], coef(fm2)[4], col = "green", lwd = 2)
text(6.8, 14, "A", cex = 1.5)
```

The form of the design matrix determines the interpretation of the parameters. Table 3.1 gives an overview of a range of linear models that can be fitted for two explanatory variables when one is a factor and the other is a covariate. We also list the case of an interaction between `pop` and `body`, and briefly review the important concept of interaction, in the next section.

3.2.2 LINEAR MODELS WITH INTERACTION BETWEEN ONE FACTOR AND ONE CONTINUOUS COVARIATE

When two terms in a linear model interact, the effect of one depends on the value of the other and vice versa. We illustrate this in the context of our ANCOVA linear model with `pop` and `body`, which, when the two interact, geometrically represents a bundle of nonparallel regression lines (Figure 3.3(b)). Interaction also means that the effects of the terms are combined in a multiplicative rather than purely additive manner. The design matrix then contains additional columns that represent the products of the columns that stand for the main effects. In a sense, the polynomial terms in Section 3.2.4 represent such interactions of a covariate with itself. We first show the effects parameterization.

```
model.matrix(~ pop*body) # Effects parameterization
  (Intercept) popAragon popCatalonia body popAragon:body popCatalonia:body
1           1         0            0  6.8            0.0               0.0
2           1         0            0  8.3            0.0               0.0
3           1         0            0  9.2            0.0               0.0
4           1         1            0  6.9            6.9               0.0
5           1         1            0  7.7            7.7               0.0
6           1         1            0  8.9            8.9               0.0
7           1         0            1  6.9            0.0               6.9
8           1         0            1  8.2            0.0               8.2
9           1         0            1  9.2            0.0               9.2
```

Here, the main effect of `pop` is represented by columns two and three, and that of `body` by column four, and the interaction terms represent the products of columns two and four and columns three and four, respectively. Next, the means parameterization.

```
model.matrix(~ pop*body-1-body) # Means parameterization
# Output slightly trimmed
  popNavarra popAragon popCatalonia popNa:body popAr:body popCat:body
1          1         0            0        6.8        0.0         0.0
2          1         0            0        8.3        0.0         0.0
3          1         0            0        9.2        0.0         0.0
4          0         1            0        0.0        6.9         0.0
5          0         1            0        0.0        7.7         0.0
6          0         1            0        0.0        8.9         0.0
7          0         0            1        0.0        0.0         6.9
8          0         0            1        0.0        0.0         8.2
9          0         0            1        0.0        0.0         9.2
```

Table 3.1 Specification of different linear models in the R and BUGS model definition languages, when `pop` is a factor, i.e., a categorical explanatory variable with three levels indexed by *j* and `body` is a covariate, i.e., a continuous explanatory variable. The factor levels of `pop` can be numbers or letters (or names) in R, but they must be integers with first level coded as 1 in BUGS. Individuals are indexed by *i* in BUGS (i.e., $i = 1...9$). Models 6 and 7 can also be reparameterised similar to the move from model 2 to model 3, but this is not shown here. The full number of model parameters is one plus the number of coefficients, because it also includes the variance.

No.	Model in R	Model in Algebra	Model in BUGS	Traditional Name of Technique Based on That Linear Model	Number of Coefficients	Meaning
1	`1`	α	`alpha`	"Model of the mean"	1	Constant term (intercept) only.
2	`pop`	$\mu + \alpha_j$	`mu + alpha[pop[i]]` `alpha[1] <- 0`	One-way ANOVA	3	This is the default linear model parameterization of a factor in R, which has an intercept (= the value for the first population) and two constants that are the differences between the values of populations 2 and 1, and 3 and 1. In BUGS, the first level of the vector alpha must be manually set to zero to avoid overparameterization. In R, this is done automatically.
3	`pop-1`	α_j	`alpha[pop[i]]`	One-way ANOVA	3	This is a simple reparameterization of model 2. There are three constants, one for each population (called the level of the factor). Called a t-test if factor has only two levels. In R, this model is specified by "subtraction" of the intercept.
4	`body`	$\alpha + \beta * x_i$	`alpha + beta * body[i]`	Simple linear regression	2	An intercept plus a slope common to all three populations (i.e., no effect of pop).
5	`body-1`	$\beta * x_i$	`beta * body[i]`	Simple linear regression through the origin	1	"Subtract the intercept": because body is a continuous covariate, this is NOT a mere reparameterization of model 4. Regression through the origin; not usually a meaningful model.
6	`pop+body`	$\alpha_j + \beta * x_i$	`alpha[pop[i]] + beta * body[i]`	Main-effects ANCOVA	4	One separate intercept for each population and a common slope.
7	`pop*body`	$\alpha_j + \beta_j * x_i$	`alpha[pop[i]] + beta [pop[i]] * body[i]`	Interaction-effects ANCOVA	6	Three separate intercepts and three separate slopes. That is, fully separate regression of wing on body for each population.

Now, the main effect of `pop` is represented by columns one through three, and the interactive effect with `body` directly by columns four through six. Let's fit this latter parameterization and plot the lines of best fit (Figure 3.3(b)).

```
summary(fm3 <- lm(wing ~ pop*body-1-body))
Call:
lm(formula = wing ~ pop * body - 1 - body)
[ ... ]
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
popNavarra          9.1296     2.7626   3.305   0.0456 *
popAragon           6.4553     3.2116   2.010   0.1380
popCatalonia        6.9217     2.9023   2.385   0.0972 .
popNavarra:body     0.1939     0.3385   0.573   0.6070
popAragon:body      0.7632     0.4077   1.872   0.1580
popCatalonia:body   0.6805     0.3559   1.912   0.1518
[ ... ]
Residual standard error: 0.5805 on 3 degrees of freedom
Multiple R-squared:  0.9992,    Adjusted R-squared:  0.9976
F-statistic: 629.9 on 6 and 3 DF,  p-value: 9.763e-05
```

```
# Plot (Figure 3.3 (b))
plot(body[sex == "M"], wing[sex == "M"], col = colorM, xlim = c(6.5, 9.5), ylim = c(10, 14),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "")
points(body[sex == "F"], wing[sex == "F"], col = colorF, pch = 16)
abline(coef(fm3)[1], coef(fm3)[4], col = "red", lwd = 2)
abline(coef(fm3)[2], coef(fm3)[5], col = "blue", lwd = 2)
abline(coef(fm3)[3], coef(fm3)[6], col = "green", lwd = 2)
text(6.8, 14, "B", cex = 1.5)
```

In BUGS, the interactive model in the effects and in the means parameterizations just shown are specified as follows:

```
for(i in 1:9){     # Effects parameterization
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- mu.alpha + alpha[pop[i]] + mu.beta * body[i] + beta[pop[i]] * body[i]
}
 alpha[1] <- 0     # Impose 'corner constraints' for identifiability
 beta[1] <- 0

 for(i in 1:9){    # Means parameterization
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- alpha[pop[i]] + beta[pop[i]] * body[i]
 }
```

We can also fit what might be called partially interactive models. We illustrate this and show how we can directly fit a model matrix using R functions such as `lm` or `glm`. Note first that the fitted regression lines in Catalonia and Aragon are rather similar, while that in Navarra is different

(Figure 3.3(a)). So let's create a design matrix for a partially interactive model with identical slopes for Aragon and Catalonia and a separate slope for Navarra, while keeping the intercepts separate for all three populations. We can conduct a significance test to compare the two models (and find out that the partially interactive model is better) and then plot the best fit-lines under this model (Figure 3.3(c)). We take the last design matrix, copy parts of the column associated with the slope in Catalonia into the slope column for Aragon, and then delete the slope column for Catalonia.

```
# Create new design matrix
(DM0 <- model.matrix(~ pop*body-1-body))  # Original DM for means param
DM0[7:9,5] <- DM0[7:9,6]                  # Combine slopes for Ar and Cat
(DM1 <- DM0[, -6])                        # Delete former slope column for Cat
  popNavarra popAragon popCatalonia popNavarra:body popAragon:body
1          1         0            0             6.8            0.0
2          1         0            0             8.3            0.0
3          1         0            0             9.2            0.0
4          0         1            0             0.0            6.9
5          0         1            0             0.0            7.7
6          0         1            0             0.0            8.9
7          0         0            1             0.0            6.9
8          0         0            1             0.0            8.2
9          0         0            1             0.0            9.2

# Fit model with partial interaction
summary(fm4 <- lm(wing ~ DM1-1))

Call:
lm(formula = wing ~ DM1 - 1)
[ ... ]
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
DM1popNavarra         9.1296     2.4018   3.801   0.0191 *
DM1popAragon          6.8230     1.8491   3.690   0.0210 *
DM1popCatalonia       6.6320     1.9106   3.471   0.0256 *
DM1popNavarra:body    0.1939     0.2943   0.659   0.5461
DM1popAragon:body     0.7162     0.2331   3.072   0.0372 *
[ ... ]
Residual standard error: 0.5047 on 4 degrees of freedom
Multiple R-squared:  0.9992,   Adjusted R-squared:  0.9982
F-statistic:  1000 on 5 and 4 DF,  p-value: 2.793e-06

# Do significance test
anova(fm3, fm4)              # F test between two models
Analysis of Variance Table

Model 1: wing ~ pop * body - 1 - body
Model 2: wing ~ DM1 - 1
  Res.Df    RSS Df  Sum of Sq      F  Pr(>F)
1      3 1.0109
2      4 1.0187 -1 -0.0078683 0.0234 0.8882
```

```
# Plot (Figure 3.3.(c))
plot(body[sex == "M"], wing[sex == "M"], col = colorM, xlim = c(6.5, 9.5), ylim = c(10, 14),
lwd = 2, frame.plot = FALSE, las = 1, pch = 17, xlab = "Body length", ylab = "")
points(body[sex == "F"], wing[sex == "F"], col = colorF, pch = 16)
abline(coef(fm4)[1], coef(fm4)[4], col = "red", lwd = 2)
abline(coef(fm4)[2], coef(fm4)[5], col = "blue", lwd = 2)
abline(coef(fm4)[3], coef(fm4)[5], col = "green", lwd = 2)
text(6.8, 14, "C", cex = 1.5)
```

In BUGS, we can use the function `inprod` to directly fit a design matrix, which must be provided to BUGS as data, see Sections 6.11.1 and 10.6. The design matrix and the parameter vector must be *compatible*, which means that the number of columns in the former must equal the length of the latter.

```
for(i in 1:9){
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- inprod(DM1[i,], beta[]) # Requires ncol(DM1) = length(beta)
}
```

The ability to directly fit a design matrix in BUGS, whether directly generated by `model.matrix` in R, or after some modifications as in our example, can be quite useful (see Sections 6.11 and 10.6).

3.2.3 LINEAR MODELS WITH TWO FACTORS

Next, we briefly look at how the effects of two categorical explanatory variables (factors) can be combined in a linear model. For one factor and one (continuous) covariate, as just shown, this is simple: the columns in the design matrix are simply joined. However, when fitting two factors, we typically have to resolve some aliasing (parameter redundancy, nonidentifiability). Put simply, aliasing means that one parameter is identical to another one or can be expressed as a linear function of other model parameters. Hence, it cannot be estimated independently. For instance, in the effects parameterization of the main-effects ANCOVA model in Section 3.2.1, the intercept μ was aliased with the three population effects. To resolve this, we set the effect of the first population (α_1) to zero; μ then became the effect of population 1 (i.e., the intercept for Navarra).

When fitting linear models in R with `lm` or `glm`, or with functions in `unmarked`, aliasing is taken care of automatically. However, when fitting linear models in BUGS, we must know which columns of the full design matrix of a model correspond to redundant parameters and then not fit them (which is equivalent to setting them to zero). This can be difficult to understand at first. For models with two or more factors that you want to fit in BUGS, it may be helpful to inspect their design matrix in R, provided you know how to specify the linear model in R. You can then directly fit the modified design matrix in R using `lm`, and in BUGS using `inprod`, as we just saw, and will see again in Sections 6.11.1 and 10.6.

Let's now see how we can fit the effects of two factors in our analysis: `pop` with three levels and `sex` with two. We consider the models that underlie the least-squares techniques known as "two-way ANOVA with main effects" and "two-way interaction effects ANOVA" (Mead, 1988).

```
model.matrix(~ pop+sex) # Design matrix of main-effects 2-way ANOVA
  (Intercept) popAragon popCatalonia sexF   # 1,3,5,7, and 9 are males
1           1         0            0    0
2           1         0            0    1
3           1         0            0    0
```

```
4          1        1           0    1
5          1        1           0    0
6          1        1           0    1
7          1        0           1    0
8          1        0           1    1
9          1        0           1    0

# Fit linear model with that design matrix
summary(fm5 <- lm(wing ~ pop + sex))
Coefficients:
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)   10.5000     0.4678  22.447  3.26e-06 ***
popAragon      1.5333     0.6375   2.405    0.0612 .
popCatalonia   1.7333     0.6125   2.830    0.0367 *
sexF           0.6000     0.5304   1.131    0.3093
---
Signif. codes:  0 '***'  0.001 '**' 0.01 '*'  0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7501 on 5 degrees of freedom
Multiple R-squared:  0.7052,   Adjusted R-squared: 0.5283
F-statistic: 3.986 on  3 and 5 DF,   p-value: 0.08536
```

The parameters associated with the four columns in the design matrix are as follows (in this order): the intercept, which is the expected wingspan of a dragonfly with level 1 for both factors (i.e., males in Navarra); the difference between a male in Aragon and one in Navarra (that's the "effect" of Aragon); the difference between a male in Catalonia and in Navarra (the effect of Catalonia); and the difference between a female and a male (the effect of sex = "F"). We do not have a separate parameter for "male" in the factor `sex`, because it would be aliased with other model parameters. Hence, it is automatically dropped from the design matrix by R.

We write the following to fit this model in BUGS and set the effects of the first levels of each factor to zero:

```
for(i in 1:9){
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- mu.alpha + alpha[pop[i]] + beta[sex[i]]
}
alpha[1] <- 0
beta[1] <- 0
```

We obtain the design matrix of the means parameterization of the same model in R as:

```
model.matrix(~ pop+sex-1) # Design matrix of the main-effects 2-way ANOVA
  popNavarra popAragon popCatalonia sexF
1          1        0           0    0
2          1        0           0    1
3          1        0           0    0
4          0        1           0    1
5          0        1           0    0
6          0        1           0    1
7          0        0           1    0
8          0        0           1    1
9          0        0           1    0
```

```
# Fit linear model with that design matrix
summary(fm6 <- lm(wing ~ pop + sex-1))
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
popNavarra     10.5000     0.4678  22.447 3.26e-06 ***
popAragon      12.0333     0.5591  21.523 4.02e-06 ***
popCatalonia   12.2333     0.4678  26.152 1.53e-06 ***
sexF            0.6000     0.5304   1.131    0.309
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7501 on 5 degrees of freedom
Multiple R-squared: 0.9978,  Adjusted R-squared: 0.996
F-statistic:  565 on 4 and 5 DF, p-value: 7.999e-07
```

Here, the parameters associated with the first three columns directly correspond to the expected wingspan of males in Navarra, Aragon, and Catalonia, respectively, and the last parameter in the mean model is the difference in wingspan between females and males. In BUGS, we can write the following (Note that `mu.alpha` from the previous parameterization of the model is dropped, which means that it is set to zero implicitly).

```
for(i in 1:9){
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- alpha[pop[i]] + beta[sex[i]]
}
beta[1] <- 0
```

Finally, let's look at some possible design matrices of a model with interaction between two factors. The following three design matrices are all different parameterizations of such an interaction-effects two-way ANOVA model. They result in separate expected wingspan estimates for the six groups, corresponding to the pairwise combination of all levels of the two factors `pop` and `sex`. Perhaps the last parameterization is the easiest to understand in terms of the meaning of the parameters: the parameters associated with columns one through three denote the expected wingspan of males in Navarra, Aragon, and Catalonia, and columns four through six denote the same for females.

```
# Variant 1: Effects parameterization (R default)
model.matrix(~ pop*sex)
#model.matrix(~ pop + sex + pop:sex)        # Same
  (Intercept) popAragon popCatalonia sexF popAragon:sexF popCatalonia:sexF
1           1         0            0    0              0                 0
2           1         0            0    1              0                 0
3           1         0            0    0              0                 0
4           1         1            0    1              1                 0
5           1         1            0    0              0                 0
6           1         1            0    1              1                 0
7           1         0            1    0              0                 0
8           1         0            1    1              0                 1
9           1         0            1    0              0                 0
```

```
# Variant 2: Means param. for pop, effects param. for sex
model.matrix(~ pop*sex-1)
  popNavarra  popAragon  popCatalonia  sexF popAragon:sexF  popCatalonia:sexF
1          1          0             0     0              0                  0
2          1          0             0     1              0                  0
3          1          0             0     0              0                  0
4          0          1             0     1              1                  0
5          0          1             0     0              0                  0
6          0          1             0     1              1                  0
7          0          0             1     0              0                  0
8          0          0             1     1              0                  1
9          0          0             1     0              0                  0

# Variant 3 (output slightly trimmed): full means parameterization
model.matrix(~ pop:sex-1)
  popNav:sexM  popAr:sexM  popCat:sexM  popNav:sexF  popAr:sexF  popCat:sexF
1           1           0            0            0           0            0
2           0           0            0            1           0            0
3           1           0            0            0           0            0
4           0           0            0            0           1            0
5           0           1            0            0           0            0
6           0           0            0            0           1            0
7           0           0            1            0           0            0
8           0           0            0            0           0            1
9           0           0            1            0           0            0
```

As an exercise, you may want to fit these linear models and compare the parameter estimates with the design matrix of the fitted model.

Variant 3 is perhaps also the easiest to code in BUGS. This shows that we can define parameters to come in multidimensional arrays: here, `alpha` is a matrix with three rows for population and two columns for sex. Thus, each combination of population and sex gets a separate parameter in the linear model.

```
for(i in 1:9){
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha[pop[i], sex[i]]   # note parameter may be a matrix
}
```

3.2.4 LINEAR MODELS WITH TWO CONTINUOUS COVARIATES AND INCLUDING POLYNOMIALS

Combinations of two or more continuous covariates are straightforward, and we illustrate them here mostly for the sake of completeness. In particular, the interaction between two covariates is simply their product. Thus, if we want to fit a model for wingspan with effects of body length and color intensity in an additive fashion, we have the following design matrix. The geometrical representation of this model is a plane in the three dimensions of wingspan, body length, and color intensity.

```
model.matrix(~ body + color) # main-effects of covariates
 (Intercept)   body  color
1          1    6.8   0.45
2          1    8.3   0.47
3          1    9.2   0.54
4          1    6.9   0.42
5          1    7.7   0.54
6          1    8.9   0.46
7          1    6.9   0.49
8          1    8.2   0.42
9          1    9.2   0.57

summary(fm7 <- lm(wing ~ body + color)) # Fit that model
Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  10.2184      3.7669    2.713    0.035 *
body          0.6579      0.4508    1.459    0.195
color        -7.5000      8.1414   -0.921    0.392
```

If we also want to fit the interaction between `body` and `color`, we get the following, the geometrical interpretation of which is a twisted surface in 3-D. The interaction term `body:color` determines the degree of twist of the surface (i.e., the deviation from a plane).

```
model.matrix(~ body*color) # Interaction between two covariates
  (Intercept) body color body:color
1           1  6.8  0.45      3.060
2           1  8.3  0.47      3.901
3           1  9.2  0.54      4.968
4           1  6.9  0.42      2.898
5           1  7.7  0.54      4.158
6           1  8.9  0.46      4.094
7           1  6.9  0.49      3.381
8           1  8.2  0.42      3.444
9           1  9.2  0.57      5.244

summary(fm8 <- lm(wing ~ body*color)) # Fit that model
Coefficients:
              Estimate  Std. Error t value Pr(>|t|)
(Intercept)     5.877     38.357   0.153    0.884
body            1.184      4.650   0.255    0.809
color           1.526     79.781   0.019    0.985
body:color     -1.088      9.555  -0.114    0.914
```

In BUGS, we can write these models as follows. For illustration, we group the regression parameters other than the intercept in the vector `beta`. We can compute the interaction between `body` and `color` as their product either inside of BUGS (variant 1) or outside in R, and then provide the product `body.color` as data (variant 2).

```
for(i in 1:9){  # Main effects
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha + beta[1] * body[i] + beta[2] * color[i]
}

for(i in 1:9){  # Interaction-effects (variant 1)
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha + beta[1] * body[i] + beta[2] * color[i] + beta[3] * body[i] * color[i]
}

for(i in 1:9){  # Interaction-effects (variant 2)
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha + beta[1] * body[i] + beta[2] * color[i] + beta[3] * body.color[i]
}
```

We often want to model covariate relationships that don't correspond to straight lines—e.g., when a response increases up to a certain covariate value and then declines again, or vice versa. Often in exploratory analyses, we may want to allow for some flexibility in the covariate effects, because we have no *a priori* knowledge about the true shape of the relationship. This is easily achieved by adding polynomial terms of a covariate (such as a quadratic or cubic) to account for deviations from a straight-line model for that covariate. We can then simply add, as explanatory variables, the square or the cube of that covariate. If we do this inside of R's `lm` function, we have to use the `I()` construct, whereas in BUGS, we use the function `pow`. The geometrical interpretation is a curvy line with one fewer inner extrema than the degree of the polynomial, with the convexity/concavity determined by the signs of the coefficients. For instance, a model with linear and quadratic terms is a parabola, which has a single extremum, while a model with cubic terms has two extrema. Note that a model with such higher-order terms should always have all the lower-order terms as well to obey the rules of marginality (McCullagh and Nelder, 1989).

```
# Cubic polynomial of body in R
body2 <- body^2       # Squared body length
body3 <- body^3       # Cubed body length
model.matrix(~ body + body2 + body3)
   (Intercept) body  body2    body3
1            1  6.8  46.24  314.432
2            1  8.3  68.89  571.787
3            1  9.2  84.64  778.688
4            1  6.9  47.61  328.509
5            1  7.7  59.29  456.533
6            1  8.9  79.21  704.969
7            1  6.9  47.61  328.509
8            1  8.2  67.24  551.368
9            1  9.2  84.64  778.688

summary(fm9 <- lm(wing ~ body + body2 + body3)) # Fit that model
# summary(fm9 <- lm(wing ~ body + I(body^2) + I(body^3))) # same
```

```
Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  165.4605   856.7892    0.193    0.854
body         -60.3906   323.7352   -0.187    0.859
body2          7.7949    40.5056    0.192    0.855
body3         -0.3306     1.6788   -0.197    0.852

Residual standard error: 1.252 on 5 degrees of freedom
Multiple R-squared: 0.1788,  Adjusted R-squared: -0.3139
F-statistic: 0.3629 on 3 and 5 DF, p-value: 0.7833
```

In BUGS, we can similarly provide the values of the higher-order polynomials of wingspan as data, or we can compute them inside of BUGS using the power function. We show the latter here.

```
for(i in 1:9){ # Cubic polynomial of body in BUGS
   wing[i] ~ dnorm(mean[i], tau)
   mean[i] <- alpha + beta1 * body[i] + beta2 * pow(body[i],2) +  beta3 * pow(body[i],3)
}
```

For relatively simple covariate relationships, polynomials are perhaps the most widely used technique, though for potentially more complex ones, you may want more flexibility still and adopt an additive model (Hastie and Tibshirani, 1990; Wood, 2006). In BUGS, you may model such very "wiggly" relationships using a mixed-model representation of penalized splines (Crainiceanu et al., 2005); see Section 10.14 for an example in the context of a logistic-regression-type of model.

3.3 GENERALIZED LINEAR MODELS (GLMs)

GLMs extend the concept of a linear effect of covariates to response variables for which a statistical distribution other than a normal must be assumed, such as the Poisson, binomial/Bernoulli, gamma, or exponential distributions. Perhaps the key feature of GLMs is that the linear effect of covariates is expressed not for the expected response directly, but for a *transformation* of the expected response (McCullagh and Nelder, 1989). That transformation is called the *link function*. Generally, we can describe a GLM for response y_i in terms of three components:

1. *Random part of the response (or the error structure* of the model)—a statistical distribution f with parameter(s) θ:

$$y_i \sim f(\theta)$$

2. *A link function* g, which is applied to the expected response $E(y) = \mu_i$, with η_i known as the *linear predictor*:

$$g(E(y)) = g(\mu_i) = \eta_i$$

3. *Systematic part of the response* (or the *mean structure* of the model): the linear predictor (η_i), which contains a linear model, such as:

$$\eta_i = \alpha + \beta * x_i$$

We can combine elements 2 and 3 and define a GLM succinctly in just two lines:

$$y_i \sim f(\theta)$$
$$g(\mu_i) = \alpha + \beta * x_i$$

Hence, a response y follows a distribution f with parameter(s) θ, and a transformation g of the expected, or mean, response is modeled as a linear function of covariates; that's the GLM in a nutshell. And this is exactly the way in which a GLM is specified in the BUGS language, and the reason why BUGS is so great for helping us to *really* understand GLMs.

The GLM concept gives you considerable creative freedom in combining the three components of a GLM, but there are typically pairs of response distributions and link functions that go together particularly well. These latter are called the *canonical* link functions. They are the *identity link* for normal responses ($\eta_i = \mu_i$), the *log link* for Poisson responses ($\eta_i = log(\mu_i)$), and the *logit link* for binomial or Bernoulli responses ($\eta_i = log(\mu_i/(1 - \mu_i))$); "identity" simply means that the expected response is not transformed, but is directly modeled as a linear function of explanatory variables. Together, these three standard GLMs make up a vast number of statistical methods used in ecology and elsewhere; for overviews, see Dobson and Barnett (2008) or Ntzoufras (2009).

The vast scope of the GLM in applied statistical analysis is one reason for the great importance of the GLM to you. A second one is that GLMs unify a very large number of seemingly unrelated models and analysis techniques; hence, understanding GLMs helps you achieve a synthetic understanding of very many techniques and models. And finally, as we will see many times in this book, GLMs can be thought of as the main building blocks for all the HMs that we develop in later chapters, including occupancy and *N*-mixture models first introduced in Chapter 2. Many exciting ecological models for inference about populations or communities can be viewed simply as a sequence of coupled GLMs (Royle and Dorazio, 2008; Kéry and Schaub, 2012).

In Section 3.2, we described what is essentially a normal-response GLM, and here we illustrate three of the most typical GLMs with nonnormal responses: Poisson, Bernoulli, and binomial GLMs. All three are of fundamental importance for the HMs in this book, because the Poisson GLM is our typical model for spatiotemporal variation of abundance, and the Bernoulli GLM plays the analogous role for occurrence ("presence/absence" or distribution data). Furthermore, in models with population dynamics (see Chapters 9 and 12—14), the Poisson is a natural model for the recruitment process, and the binomial for the survival process. Finally, our canonical description of the observation process governed by false-negatives is the Bernoulli or the binomial distribution (Royle and Dorazio, 2008; Kéry and Schaub, 2012), although sometimes a Poisson, or a Bernoulli that *implies* an underlying Poisson process, may be more adequate; see Efford et al. (2013), Royle et al. (2014), and also Section 3.3.6 and Exercise 4.8 in Chapter 4. If you understand Poisson and Bernoulli GLMs, you are in good shape to understand all the HMs in this book.

Technically, GLMs are defined for all members of statistical distributions belonging to the so-called exponential family (McCullagh and Nelder, 1989; Dobson and Barnett, 2008; Ntzoufras, 2009), which includes the normal, Poisson, binomial/Bernoulli, multinomial, beta, gamma, lognormal, exponential, and Dirichlet distributions. Thus, the principles of linear modeling can be carried over to a vast number of "error models" other than the normal—i.e., to a very large number of stochastic processes that produce random outcomes that are predictable in some average sense only.

3.3.1 POISSON GLM FOR UNBOUNDED COUNTS

The Poisson distribution is the number of “things” from a Poisson process, which is a natural model for independent discrete “things” indexed by space or time (Gelman et al., 2014). “Things” is a placeholder for much of what scientists usually count. A Poisson GLM is a natural choice for unbounded counts—i.e., nonnegative integers that have no logical upper bound (unlike binomial counts, which cannot be greater than their sample size; see Section 3.3.7). We will use the parasite load in our toy dragonfly data set for illustration. For the number of mites counted on dragonfly i, C_i, we can write the main-effects ANCOVA linear model for factor `pop` and the body length covariate `body` within a Poisson GLM as follows:

1. Random part of the response (statistical distribution for the randomness in the response):

$$C_i \sim Poisson(\lambda_i)$$

2. Link of random and systematic part in response (link function):

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

3. Systematic part of the response (linear predictor η_i for link-transformed mean response):

$$\eta_i = \alpha_j + \beta * body_i$$

Here, λ_i is the expected (or mean) response for dragonfly i on the arithmetic scale ($E(C)$), η_i is the expected count on the log link scale, also called the linear predictor, $body_i$ is the body length of dragonfly i, and α and β are the two parameters (α being vector-valued) of the log-linear regression of the counts on factor `pop` and covariate `body`.

We fit this model in R and give the BUGS code for fitting it. Once again, we will see that the model descriptions in algebra and in the BUGS language are essentially identical. Here is the model in algebra in short; note that C corresponds to the variable `mites` in our toy data.

$$C_i \sim Poisson(\lambda_i)$$

$$\log(\lambda_i) = \alpha_j + \beta * body_i$$

```
summary(fm10 <- glm(mites ~ pop-1 + body, family = poisson))

Call:
glm(formula = mites ~ pop - 1 + body, family = poisson)
[ ... ]
Coefficients:
               Estimate  Std. Error  z value  Pr(>|z|)
popNavarra      -6.634      2.325   -2.854   0.00432 **
popAragon       -5.878      2.216   -2.653   0.00799 **
popCatalonia    -5.519      2.290   -2.410   0.01596 *
body             0.845      0.261    3.237   0.00121 **
[ ... ]
```

```
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 59.658  on 9  degrees of freedom
Residual deviance: 15.467  on 5  degrees of freedom
AIC: 42.591

Number of Fisher Scoring iterations: 6
```

We can specify the linear model for the factor `pop` in the means or effects parameterization, exactly as before. The methods for fitting and interpreting covariates, including two factors such as `pop` and `sex`, are exactly analogous for linear models and GLMs. To specify a Poisson GLM in the BUGS language using the means parameterization, we write:

```
for(i in 1:9){
  mites[i] ~ dpois(lambda[i])
  log(lambda[i]) <- alpha[pop[i]] + beta * body[i]
}
```

This is *exactly* how we just wrote the model in algebra, where `pop[i]` now takes the place of the index j for the intercept α; i.e., α_j in algebra is written as `alpha[pop[i]]` in the BUGS language.

3.3.2 OFFSETS IN THE POISSON GLM

The expected count C of a Poisson response is λ. Often we have a systematic component in the variation of C that is due to known variation in the size of the spatial or temporal unit, or effort, for which a count was obtained. For instance, the Poisson counts may have been obtained on sample plots of varying size, along transects of varying length or over temporal "observation windows" of varying duration. If we denote the size of the observation window as A, then all else equal it would be natural to expect a count of A times λ, where λ is the expected count per unit-sized observation window. For instance, if we expect 2 peregrine falcon pairs in a randomly placed 100 km^2 study plot in the Jura mountains, then we would probably expect 4 pairs for a study plot twice that size. When modeling variability in the expected count, we will then do this for $A * \lambda$ on the log scale. We illustrate the principle here for a simple loglinear regression on a covariate x:

$$C_i \sim Poisson(A_i * \lambda_i)$$
$$\log(A_i * \lambda_i) = \log(A_i) + \log(\lambda_i) = \log(A_i) + \alpha + \beta * x_i,$$

where the natural logarithm of A, $\log(A_i)$, is called an offset. An offset can be described as a covariate with coefficient fixed at 1, hence we can also describe the linear predictor as:

$$\log(A_i * \lambda_i) = \alpha + 1 * \log(A_i) + \beta * x_i$$

One consequence of adding an offset to a Poisson GLM is that now λ has the interpretation of a *density* relative to the units in which A is measured. The implied coefficient of 1 for the offset term in the linear model means that we expect density to be independent of the area. We could test this assumption by actually estimating a coefficient for $\log(A_i)$: if is greater than 1, density becomes greater in larger than in smaller areas, and vice versa.

Typically, the value of the offset variable A will be greater than 1, but sometimes it may be less than 1 (but greater than 0). For instance, we can specify an offset equal to log(p) to accommodate a known value of imperfect detection p in an *ad hoc* (two-step) HM for counts (Hedley and Buckland, 2004; Oedekoven et al., 2013; Solymos et al., 2013). With imperfect detection, we can write a count C as $C \sim Poisson(\lambda p)$; i.e., the expected count $E(C)$ is the product of the expected local abundance λ, and detection probability p. If we want to model variation in the expected abundance λ, while accounting for a known value of imperfect detection p (which will typically be an estimate coming from another analysis), we can express the log of the expected count as:

$$\log(E(C)) = \log(\lambda p) = \log(\lambda) + \log(p)$$

i.e., as the sum of the log(expected abundance) and log(detection probability). To specify models of the usual form for among-site variation in the expected abundance, we can then write (for the example of a loglinear regression of density λ at site i on a covariate x):

$$\log(E(C_i)) = \beta_0 + \beta_1 * x_i + \log(p_i)$$

Now, the coefficients β_0 and β_1 relate the covariate x to the expected abundance $E(N)$. This type of analysis assumes implicitly that density and detection probability are not affected by the same covariates. This is a drawback of a two-step analysis that is not shared by a joint model where both abundance and detection are formally estimated in a single hierarchical model, as in Royle et al. (2004) in the context of a distance sampling model (see also Chapters 8 and 9).

As an example for an offset in our dragonfly data set, let's standardize the mite counts by body size and regress it on another measure of size, wingspan. In this way, we are effectively modeling mite density per unit body length.

```
summary(fm10a <- glm(mites ~ pop-1 + wing, offset = log(body), family = poisson))
# summary(fm10a <- glm(mites ~ offset(log(body)) + pop-1 + wing, family = poisson))
# same

Call:
glm(formula = mites ~ pop - 1 + wing, family = poisson, offset = log(body))

Coefficients:
               Estimate   Std. Error    z value   Pr(>|z|)
popNavarra    -21.3736      6.7968      -3.145    0.00166 **
popAragon     -25.0056      8.3933      -2.979    0.00289 **
popCatalonia  -23.9713      8.1046      -2.958    0.00310 **
wing            1.8379      0.6271       2.931    0.00338 **
```

We see that mite density per unit body length increases significantly with wingspan. We also see that no coefficient is estimated for the offset term, since it is fixed at 1.

To specify in BUGS an offset for body length in a GLM of mite count on wingspan, we simply specify the linear model in one of the following ways:

```
for(i in 1:9){
   mites[i] ~ dpois(lambda[i])
   log(lambda[i]) <- log(body[i]) + alpha[pop[i]] + beta * wing[i]
#  log(lambda[i]) <- 1 * log(body[i]) + alpha[pop[i]] + beta * wing[i] # same
}
```

3.3.3 OVERDISPERSION AND UNDERDISPERSION

Neither the Poisson nor the binomial distributions have a dispersion parameter to govern the variability of the response. Rather, the magnitude of that variability is a fixed function of the mean: the variance is equal to the Poisson mean λ, equal to $Np(1 - p)$ for the binomial and equal to $p(1 - p)$ for the Bernoulli. Very often, we observe what is called overdispersion: that the variance is greater than expected from these expressions. In the Poisson, a simple index of overdispersion is the variance/mean ratio, which should be about 1. Note, however, that for the raw data this index is solely applicable for an intercepts-only model. The model's dispersion is what is "left over" after any structure in the mean response due to covariates has been accounted for. Hence, it would be wrong to observe a variance/mean ratio in the raw data that is greater than 1 and then conclude that a Poisson GLM was inadequate: it may well be that the "overdispersion" can be explained adequately with the available covariates. Overdispersion must be gauged *after* any known structure in the mean response has been accommodated in the model. Therefore, if we fit a Poisson model with covariates in R, we must inspect the ratio of the residual deviance to the residual degrees of freedom, and only then does a ratio substantially greater than 1 indicate overdispersion. The analogous applies to a binomial response (but note that overdispersion cannot be measured or modeled for a Bernoulli response; McCullagh and Nelder, 1989).

Overdispersion typically arises when some structure in the mean response is ignored. For instance, important covariates may have been left out of the model or hidden structure is ignored—e.g., due to clustering of individuals in groups, or due to spatial or temporal correlation. The result of overdispersion is usually that uncertainty assessments are too optimistic, because standard errors are too small and confidence intervals too short. It is important to accommodate this feature of the response. For instance, you could explicitly model such remaining structure in the expected response by adding these covariate effects or some random effects accommodating the missing grouping structure in your data. However, in practice you won't always have measured or even know them all, so various remedies are employed that simply measure the amount of overdispersion and adjust the uncertainty assessments accordingly. In frequentist analyses, quasi-likelihood is commonly employed, whereby it is assumed that the variance of the data differs from the, say, Poisson variance by a constant and estimable overdispersion factor ϕ (McCullagh and Nelder, 1989). This can also be done in more complex models; see for instance Johnson et al. (2010) for an application of overdispersion correction in a hierarchical distance sampling model (for which you may see Chapters 8 and 9). A more explicitly model-based alternative is to fit an additional zero-mean random effect that soaks up unexplained variation and in effect then acts as an additional variance parameter that is lacking in the nominal Poisson or binomial. When we fit a binomial mixture model with Poisson-lognormal distribution for abundance (see Section 6.11), we do just that. As an alternative to a Poisson, we could also adopt a negative binomial distribution that allows for extra-Poisson variability through its dispersion parameter. Indeed, a number of more flexible alternatives to the Poisson exist (e.g., see Wu et al., 2013, 2015).

Actually, "overdispersion" in the wider sense may sometimes be observed with almost any standard distribution including the normal, Poisson, and binomial, and all three have some classic alternatives that may be adopted in this case and allow for more variability (or more extreme values in the normal). For instance, when there are too many outliers in a normal model, a t-distribution is a more robust alternative (Jonsen et al., 2005). The classical overdispersed Poisson alternative is the negative binomial and an overdispersed alternative to the binomial is the beta-binomial (Gelman et al., 2014; Lynch et al., 2014). All of these can be fitted in BUGS.

The converse to overdispersion is underdispersion. This is much more rarely observed, and the usual advice is to simply ignore it. However, there may be an interest in accounting for the reduced variability in a response, because otherwise our uncertainty assessments will be too pessimistic (e.g., SEs become too wide). One example in population ecology is avian clutch size, which for most species varies somewhere between 1 and 10, and there is some upper limit that is very rarely exceeded. A Poisson with variance equal to the mean observed clutch size will frequently allow unrealistically high clutch sizes. One way of accounting for this reduced variability would be an adaptation to a Poisson (see Ridout and Besbeas, 2004; also Wu et al., 2013, 2015). A simple alternative in BUGS is to fit a binomial instead, where the trial size N is set equal to the highest clutch size ever observed. To model pattern in the mean clutch size, we can use *moment matching* (Hobbs and Hooten, 2015) and model log(Np) as a linear function of covariates. See Section 5.12 and also 7.6 for an example of moment matching in the modeling of a negative binomial response.

3.3.4 ZERO-INFLATION

One particular way in which an observed response can have a different dispersion structure than under the nominal distribution is called zero-inflation: there are more zeros than expected under the nominal distribution. This is very common for count data, but may also occur for continuous measurements. Zero-inflation can be thought to arise as a consequence of two linked processes, the first of which generates structural zeros. For instance, when modeling abundance with a Poisson distribution or species occurrence with a Bernoulli, there may be sites that are unsuitable in principle. These sites will necessarily produce zero abundance, beyond the sampling zeros arising from the random process represented by the Poisson or the Bernoulli distribution. Thus, a zero-inflated Poisson (ZIP) distribution for abundance N can be described as follows

$$s_i \sim Bernoulli(\theta) \quad \text{\# s is binary indicator of site suitability}$$

$$N_i \sim Poisson(s_i \lambda) \quad \text{\# model for realized abundance}$$

Only when a site is suitable (i.e., $s_i = 1$) can abundance be greater than zero, while when $s_i = 0$, we have $N_i \sim Poisson(0)$ and will only ever observe zero abundance. Under this model, zeros can arise from both processes: we naturally get $N_i = 0$ when a site is unsuitable (i.e., $s_i = 0$), but we may also obtain a zero by chance when rolling our "Poisson dice" for a site that is suitable in principle, where $s_i = 1$. Thus, it would be wrong to say that sites with zero abundance are unsuitable under this model. We find this a perfectly natural way of thinking about zero inflation arising due to a suitability process, but some don't like the fact that zero abundance can arise from both processes. An alternative is the use of hurdle models (e.g., Potts and Elith, 2006; Dorazio et al., 2013), which adopt a zero-truncated Poisson in the second (abundance) part of the HM. Hurdle models have zero abundance coming only from the suitability process and hence, sites with zero abundance can rightly be interpreted as unsuitable.

Similarly, when modeling occurrence in the presence of zero-inflation, we have

$$s_i \sim Bernoulli(\theta) \quad \text{\# s is binary indicator of site suitability}$$

$$y_i \sim Bernoulli(s_i p) \quad \text{\# model for observed presence/absence}$$

Such zero-inflated GLMs may readily be fitted in R using functions in package `pscl` (Jackman, 2012), and they are easy to implement in BUGS as an HM exactly analogous to the above expressions (Zuur et al., 2012). Indeed, the site-occupancy model (Chapter 10) is a special kind of zero-inflated binomial (ZIB) model, where sites at which a species is absent inflate the number of zeros in the observed detection/nondetection observations (Tyre et al., 2003).

Zero-inflated models are a powerful way to accommodate a typical feature of count data: that there often are too many zeros relative to a nominal distribution such as the Poisson or even the negative binomial. Moreover, it may sometimes make sense to imagine two linked processes that govern spatial variation in abundance: one process governs the suitability of a site and thus indirectly affects abundance by throwing into the abundance data structural zeros for *N*. It also seems sensible to assume that different environmental covariates affect the suitability and the abundance processes underlying the realized abundance. However, in a standard ZIP model with a single datum per site, we would strictly advise against using the same covariates in both parts of a zero-inflated model, since it seems this would induce some sort of nonidentifiability in the associated parameters. In contrast, for a ZIB model with multiple detection/nondetection observations available per site (i.e., the classical setting for a site-occupancy or an *N*-mixture model), this is perfectly fine since the two parts of the ZIB model really do represent two entirely distinct processes, and we have the information to differentiate between them. See also Section 1.1 for a general discussion of zero-inflated models and whether we can hope to find much biology in the zero-inflated part of such models.

3.3.5 BERNOULLI GLM: LOGISTIC REGRESSION FOR A BINARY RESPONSE

We use a Bernoulli distribution to describe patterns in binary responses—i.e., the outcomes of coin flip-like random processes. The Bernoulli distribution is also important because perhaps the more commonly encountered binomial distribution with trial size *N* is the sum of the outcomes of *N* independent Bernoulli trials. In this book, the Bernoulli (or alternatively, the binomial; see Section 3.3.7) is the canonical description of the measurement error process involved in all ecological studies of distribution or abundance, where the presence of a species or of an individual can be overlooked with probability equal to 1 minus the detection probability. That is, the probability of measurement error is the converse of detection probability. Unlike in most sciences where measurement error is typically modeled as a continuous random process (for instance using a normal distribution), for the measurement of distribution or abundance, measurement error is a binary process at the scale of each individual or the presence/absence state of species at a site. See Section 1.4 for an introduction to measurement error in population ecology.

For our dragonfly example data set, we can define a binary random variable, where the state "parasite presence" corresponds to a parasite load greater than zero. The event of a dragonfly with "parasite presence" is a so-called "success" (albeit less so for the dragonfly) and coded as 1, while the converse, "parasite absence," is traditionally called a "failure" and coded as 0. We call *y* the indicator for a parasite load greater than zero; it contains a 1 if a dragonfly has at least one parasite and a 0 if it is parasite-free. Our Bernoulli GLM for the occurrence of at least one parasite on a dragonfly in relationship to population and body length is then:

$$y_i \sim Bernoulli(p_i)$$

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \alpha_j + \beta * body_i$$

Here, p_i is the expected proportion of dragonflies with presence of parasites on the probability scale—i.e., with a parasite load of one or greater. This is the mean response at the level of each individual dragonfly. The linear model is applied to the logit transformation of that expected response. Thus, in a Bernoulli GLM we are modeling a proportion or probability.

```
presence <- ifelse(mites > 0, 1, 0) # convert abundance to presence/absence
summary(fm11 <- glm(presence ~ pop-1 + body, family = binomial))
Call:
glm(formula = presence ~ pop - 1 + body, family = binomial)
[...]
Coefficients:
              Estimate  Std. Error   z value  Pr(>|z|)
popNavarra     -17.263     10.974    -1.573     0.116
popAragon      -16.710     10.557    -1.583     0.113
popCatalonia   -17.273     10.948    -1.578     0.115
body             2.297      1.416     1.622     0.105

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12.4766  on 9  degrees of freedom
Residual deviance:  6.6209  on 5  degrees of freedom
AIC: 14.621

Number of Fisher Scoring iterations: 5
```

There is nothing new here in terms of the model specification in R. To specify this model in BUGS, we again write almost the identical thing as in algebra:

```
for(i in 1:9){
  presence[i] ~ dbern(p[i])
  logit(p[i]) <- alpha[pop[i]] + beta * body[i]
}
```

The logit link is the typical link employed in binomial or Bernoulli GLMs. Other link functions include the probit and the complementary log-log (cloglog), which are both available in R and BUGS. They give very similar results especially in the range of $0.1 < p < 0.9$, although the cloglog is asymmetric about 0.5. In custom-made MCMC algorithms, the probit can enjoy considerable efficiency benefits (Dorazio and Rodriguez, 2012; Johnson et al., 2013). The cloglog link leads to an astonishing interpretation of presence/absence data in terms of an underlying abundance; we describe this next.

3.3.6 MODELING A POISSON PROCESS FROM PRESENCE/ABSENCE DATA USING A BERNOULLI GLM WITH CLOGLOG LINK

The cloglog link function may sound obscure to most ecologists, but it has an intriguing feature: it effectively expresses the binomial success probability as a function of the intensity of an underlying

Poisson process (Royle and Dorazio, 2008, p. 150; Baddeley et al., 2010). Thus, under suitable distributional assumptions of a Poisson point process, a cloglog Bernoulli GLM enables us to recover abundance estimates from mere presence/absence data! To see why, consider the fundamental relationship between occurrence (z) and abundance (N): we have an occurrence, or presence ($z = 1$), whenever abundance is greater than zero. Hence, for occupancy probability ψ we can write:

$$\psi = \text{Prob}(z = 1) = \text{Prob}(N > 0) = 1 - \text{Prob}(N = 0)$$

This expression is based on the fundamental relationship between occurrence and abundance, and does not depend on any statistical distribution assumed for N. If we assume a Poisson distribution with intensity λ, we have $\text{Prob}(N = 0) = \exp(-\lambda)$, and hence occurrence probability ψ is related to the expected abundance λ as follows:

$$\psi = 1 - \exp(-\lambda)$$

Adopting a cloglog link for occurrence probability ψ is equivalent to adopting a log link for λ, because we can rearrange the above expression to $\lambda = -\log(1 - \psi)$, and, taking logs, we get

$$\log(\lambda) = \log(-\log(1 - \psi)),$$

which is exactly the cloglog link for ψ. Hence, applying the cloglog link to the Bernoulli parameter ψ is equivalent to linear modeling of $\log(\lambda)$ for the intensity parameter of an underlying Poisson abundance model! We find this fascinating, since under suitable distributional assumptions, it allows you to estimate and model the abundance underlying an observed presence/absence process. We illustrate this for the occurrence models for mites in dragonflies by fitting the two equivalent models.

	Model for Mite Presence/Absence	**"Direct" Model for Mite Count**
Distribution	$presence_i \sim Bernoulli(\psi_i)$	$mites_i \sim Poisson(\lambda_i)$
Linear predictor	$\text{cloglog}(\psi_i) = \alpha_j + \beta * body_i$	$\log(\lambda_i) = \alpha_j + \beta * body_i$

Let's try this in practice and fit the "implicit Poisson model" to the presence/absence data via a Bernoulli GLM with cloglog link function.

```
summary(fm11a <- glm(presence ~ pop-1 + body, family = binomial(link = "cloglog")))
Coefficients:
            Estimate   Std. Error   z value  Pr(>|z|)
popNavarra   -12.447       8.545    -1.457     0.145
popAragon    -12.509       8.216    -1.523     0.128
popCatalonia -12.410       8.459    -1.467     0.142
body           1.610       1.071     1.504     0.133
```

We compare with the Poisson model for the mite counts directly (this is `fm10`).

```
summary(fm10)
Coefficients:
             Estimate Std. Error   z value  Pr(>|z|)
popNavarra    -6.634     2.325    -2.854   0.00432 **
popAragon     -5.878     2.216    -2.653   0.00799 **
popCatalonia  -5.519     2.290    -2.410   0.01596 *
body           0.845     0.261     3.237   0.00121 **
```

Admittedly, this is perhaps not too convincing: the coefficients of the two models don't agree very well. However, this should not be too surprising, since our sample is extremely small and we did not generate the mite counts using a Poisson distribution. In Exercise 4.8 in Chapter 4, you will see a much more convincing example of how the parameters of an underlying Poisson process can be estimated from simple presence/absence data using a Bernoulli GLM with a cloglog link.

In the BUGS language, a cloglog Bernoullli GLM is specified in either of the following:

```
for(i in 1:9){
  presence[i] ~ dbern(psi[i])
  psi[i] <- 1 - exp(-alpha[pop[i]] - beta * body[i])
}


for(i in 1:9){
  presence[i] ~ dbern(psi[i])
  cloglog(psi[i]) <- alpha[pop[i]] + beta * body[i]
}
```

Bernoulli GLMs with a cloglog link are important because they provide the natural link between occurrence and abundance, if the latter is approximately Poisson. Moreover, this model shows how occupancy probability at sites of different areas can be reconciled, namely by adoption of a complementary log-log Bernoulli regression with the logarithm of the area of the grid cells used as an offset (Baddeley et al., 2010; Dorazio, 2014; Fithian et al., 2014).

3.3.7 BINOMIAL GLM: LOGISTIC REGRESSION FOR BOUNDED COUNTS

The binomial distribution is the standard model for counts of independent and identically distributed discrete "things" that come in one of two types and which have a fixed total number (Gelman et al., 2014). As for the Poisson, "things" is again a very generic placeholder. Type refers to binary distinctions such as "presence/absence," "male/female," "dead/alive," "here/away," "reproductive/nonreproductive" or "detected/nondetected." The binomial distribution is adopted as a model for random variables that can be considered a sum of N independent Bernoulli trials, where N is called the sample or trial size. Typical examples may be the number of female nestlings in a nest with N nestlings, individuals parasitized in a group of N individuals scored for some parasite, individuals dying over a time period in a group of N individuals, or individuals detected among those present in a population of size N. A binomial count can never exceed the sample size, and hence it is naturally bounded. This is a major distinction from Poisson counts. Note also that the binomial is *not* an adequate model for every ratio of two numbers. For instance, it would not generally be adequate for a ratio of two areas—e.g., when your interest lies in the proportion of the leaf area chewed by some beetle. The binomial is a model for the ratio of two counts only.

We will see binomial GLMs many times throughout this book, since they or the elemental Bernoulli GLMs are our canonical descriptions of the observation process for distribution, abundance and species richness. At the level of the observed data, most models in this book have a binomial distribution; hence, there is a sense in which they can all be considered logistic regressions, albeit owing to latent variables such as presence or abundance, they typically have some fairly complicated random-effects structures.

For our toy dragonflies, we model as a binomial count the number of damaged wings in individual *i* from among the four wings (variable $damage_i$), in relation to population and body length:

$$damage_i \sim Binomial(N, p_i)$$
$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \alpha_j + \beta * body_i$$

Here, $N = 4$ is the value of the binomial sample size, corresponding to the four wings. At the elemental level of a single wing, the expected response of this binomial (or Bernoulli) count is p_i; it is the expected proportion of damaged wings of an individual on the probability scale. Again, from observing counts, we model an underlying probability. The linear model is applied to the logit transformation of that expected response or probability. Note, however, that the expected value of *damage* is $N * p$, or the sum of *N* Bernoulli trials each with probability *p*. In R, the response of a binomial GLM with sample size greater than one (i.e., not Bernoulli) is specified as the pair (number of successes, number of failures).

```
summary(fm12 <- glm(cbind(damage, 4-damage) ~ pop + body -1, family = binomial))

Call:
glm(formula = cbind(damage, 4 - damage) ~ pop + body - 1, family = binomial)
[...]
Coefficients:
              Estimate  Std. Error   z value  Pr(>|z|)
popNavarra     -4.7643      3.9351    -1.211     0.226
popAragon      -3.0047      3.6658    -0.820     0.412
popCatalonia   -4.7601      3.9255    -1.213     0.225
body            0.3837      0.4623     0.830     0.407

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 29.819  on 9  degrees of freedom
Residual deviance: 17.457  on 5  degrees of freedom
AIC: 32.833

Number of Fisher Scoring iterations: 4
```

In BUGS, a binomial GLM is specified exactly as in algebra above, except for the order of the arguments: success probability comes before the binomial sample size:

```
for(i in 1:9){
  damage[i] ~ dbin(p[i], 4)
  logit(p[i]) <- alpha[pop[i]] + beta * body[i]
}
```

3.3.8 THE GLM AS THE QUINTESSENTIAL STATISTICAL MODEL

There can be very few things in nature that are entirely predictable in space or time. Hence, essentially everything, in science and in life, has a chance element in it. As we have seen in Chapter 2, statisticians use probability to describe this element of chance and invoke the concept of a random variable. This is a hypothetical random process described by a probability density function. We imagine that the random variable has produced our observed data set, which we have an interest in explaining. The most common density functions include the normal, Poisson, and binomial distributions. A GLM simply represents the idea of adopting linear models for a link transformation of the expected response for certain types of random variables. Hence, the GLM is the quintessential statistical model—i.e., a formal description of a defined random process that produces some result that we are interested in explaining, describing, or predicting. The magnitude of the unpredictable part of the response is accounted for by the inherent variability of the random variable (this is the stochastic part of the response), while the average behavior of the response (its deterministic part) is accommodated by the linear model.

3.4 RANDOM EFFECTS (MIXED) MODELS

We previously defined an HM as a series of probability models for linked random variables where the outcome of one random variable is observed (this is the data), and the outcome of one or more additional random variables is not observed (or only partially so) and is thus latent. The outcomes of these entirely or at least partially latent random variables are called *random effects*. As we argued in Chapter 2, in HMs in this book, random effects arise naturally under a mechanistic view of the stochastic processes that underlie the observed data on occurrence and abundance in site-structured populations. A population at site i has a certain state (e.g., a species is present or absent, or we have abundance state N_i) that is observed imperfectly due to the peculiar, binary kind of measurement error induced by imperfect detection (and possibly also due to a false-positive error component). The latent occurrence or abundance state of a local population is viewed as the outcome of a stochastic process whose features we want to model; for instance, we want to express the occurrence probability or the expected abundance as a function of environmental covariates. Hence, it is natural to treat the population state as a random effect governed by some probability distribution. Notably, this type of random effect has a clear ecological meaning as one of the fundamental states in ecology: occurrence (or the element of a species distribution) and abundance.

In contrast, in the vast majority of examples of random-effects (or mixed) models in ecology, the random effects do *not* have a clear ecological interpretation. Rather, they are merely abstract constructs invoked to explain the fact that some measurements are more similar to each other than others are—i.e., to model correlations in the observed data. For instance, in a randomized-block ANOVA model, random "group effects" are introduced into the linear predictor for a response, to accommodate the fact that measurements taken in the same block or group all have a tendency to be higher or lower than the overall average (Mead, 1988; Littell et al., 2008). The shared group effect induces a correlation in the responses, and is treated as the outcome of a random process typically described by a normal distribution. Thus, random effects are effects, or parameters, that one may or may not want to estimate individually, and that are given a (prior) distribution (e.g., a normal) that itself has (hyper) parameters, such as the mean and the variance, that are estimated. We make the assumption that the group effects are *exchangeable*, which for practical purposes means independent and identically distributed.

The motivations to declare a set of effects as random—i.e., as the outcome of a stochastic process—are many and varied; see Kéry and Schaub, 2012, p. 77–82. We briefly summarize in the context of our dragonfly example from Sections 3.2.1 and 3.2.2, as far as possible:

- *Increased scope of inference and assessment of variability*: Treating the effects of population as random allows us to make an inference not only about the three particular populations sampled, but also about an entire "population of populations" from which the three populations are considered to be a sample. For instance, we can estimate the mean and the variability among the populations represented by our sample of three; these are the hyperparameters of the random effects prior.
- *Accounting for hidden structure in the data*: The classical example is the block effect in a randomized block ANOVA or in similar random-effects one-way ANOVA models (Mead, 1988; Chapter 9 in Kéry, 2010). Accounting for such hidden structure accommodates the correlations that exist and prevents pseudoreplication (Hurlbert, 1984).
- *Partitioning of variability*: Not only can we estimate the variability among populations, but we may also start to explain the differences among populations by measured covariates. Thus, whenever we want to *model* some parameters as a function of covariates, it is natural to assume that they are the outcome of a random process (i.e., random effects), the hyperparameters of which we can then model as being affected by those explanatory variables, by adopting a linear model at this higher level of the model; see Section 11.6.3 and Figure 11.12 for an example of this.
- *Modeling of correlations among parameters*: Not only can we model variability among a single set of parameters (e.g., population effects), but we can also model underlying correlations among two or more sets of parameters, such as survival and reproduction (Cam et al., 2002a; Schaub et al., 2013), juvenile and adult survival over time (Kéry and Schaub, 2012, p. 204–208) or pairs of values for occupancy and detection probability in a community occupancy model; see Section 11.6.2 for an example of how to code this in the BUGS language.
- *Modeling of spatial, temporal, or spatiotemporal correlation*: Spatial or temporal autocorrelation can be accommodated by adding correlated spatial or temporal random effects, with the correlation being a function of their distance in space or time (Banerjee et al., 2004; Cressie and Wikle, 2011; also see Chapter 21).
- *Partial pooling, borrowing strength, and "shrinkage"*: Treating population effects as random can be seen as a compromise between assuming all populations are equal (complete pooling of effects; corresponding to a model without `pop` effects at all) on the one hand, and assuming that they are totally unrelated (no pooling) on the other (Gelman, 2006; Gelman and Hill, 2007), corresponding to a model with fixed `pop` effects. By treating the effects as exchangeable—i.e., as similar but not identical—the estimate of each population is affected not only by the dragonflies measured in it, but also by the other six dragonflies measured in the other two populations. In this way, the estimate for each population "borrows strength" from the ensemble of populations. This means that random-effects estimates of `pop` will *not* be the same as fixed-effects estimates; rather, they will be pulled in towards their overall mean, and this is called "shrinkage" (Carlin and Louis, 2009; see Figure. 3.4).
- *Combining information*: Treating a set of parameters as random is a way of combining their information. For instance, parameters may be estimates from different studies. Treating them as random allows one to estimate their mean hyperparameter, which is a single measure of effect size that combines the information from all the studies (McCarthy and Masters, 2005; Schaub and Kéry, 2012). This is also called a meta-analysis in certain settings.

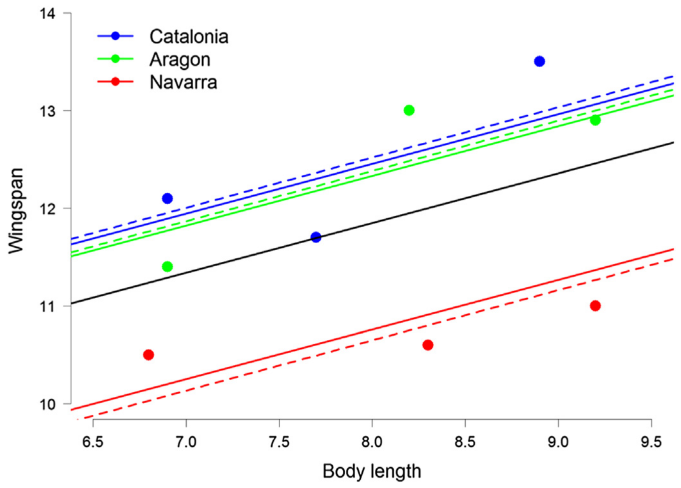

FIGURE 3.4

Comparison of lines of best fit under a `pop+body` linear model with fixed `pop` effects (dashed lines) and random `pop` effects (solid lines), respectively. The fixed `pop` effects (α_j) are estimated as three completely unrelated parameters, while when assumed random, they are estimated subject to the additional assumption $\alpha_j \sim Normal(\mu_\alpha, \sigma_\alpha^2)$—i.e., as related quantities. The random-effects lines are pulled in ("shrunk") towards the grand mean, μ_α, which is represented by the solid black line.

Treating sets of parameters as random has many advantages, but should perhaps not be done by default (though see Gelman, 2005): if the assumption of exchangeability does not hold, we will mix apples and oranges and may obtain meaningless estimates of hyperparameters, and hence random-effects estimates that are shrunk towards a nonsensical overall mean. Further, if interest lies in measuring the variation among random effects, a certain number is required; for instance, it does not make much sense to estimate a variance among populations in our dragonfly example with only three populations; we simply do this for illustrative purposes here. To obtain an adequate estimate of the among-population heterogeneity—that is, the variance parameter—at least 5—10 populations might be required. And finally, random-effects models are typically computationally (much) more expensive, both in frequentist and in Bayesian modes of analysis. Hence, there may be good reasons to not always treat all factors as random.

In the remainder of this chapter, we illustrate traditional random- or mixed-effects models, first for a normal and then for a Poisson response. These are mixed-effects models because they contain both random and fixed effects, and we call them traditional because they represent the typical motivation for random effects, to account for some hidden structure in the data: by treating the population effects as random, we account for correlated measurements among dragonflies in the same population. These population effects are not something with a clear ecological interpretation, but rather some elusive tendency for some populations to be above and others to be below the population average. Since these random population effects are continuous quantities, we assume a normal distribution for them. This is what is done in the vast majority of mixed-effects models in ecology and is also the default and only option in major standard software to fit such traditional mixed models, such as function `lmer` in the R package `lme4` (Bates et al., 2014). However, as we have seen in the site-occupancy and *N*-mixture models in Chapter 2, random effects may be discrete and have distributions other than a normal; for instance, a Bernoulli or Poisson, respectively.

3.4.1 RANDOM EFFECTS FOR A NORMAL DATA DISTRIBUTION: NORMAL-NORMAL GLMM

The distribution assumed for the observed random variable—i.e., the data—is often called the data distribution or observation model. We will use our toy dragonfly example to illustrate a random-effects version of the linear regression model, and especially shrinkage; compared with their fixed-effects counterparts, random-effects estimates are pulled in (or "shrunk") towards the mean of their prior distribution. Here, we revisit the ANCOVA example from Section 3.2.1 and recreate the figure from before, but now without keeping track of the sex of the nine dragonflies. This is a normal-normal mixed model or an HM, because the distributions of the data and the random population effects are both normal.

```
# Plot data without distinguishing sex
plot(body, wing, col = rep(c("red", "blue", "green"), each = 3), xlim = c(6.5, 9.5), ylim =
c(10, 14), cex = 1.5, lwd = 2, frame.plot = FALSE, las = 1, pch = 16, xlab = "Body length", ylab =
"Wingspan")
```

The model adopted in Section 3.2 assumed a linear relationship between wingspan and length, a different baseline in each population, and residuals ε_i coming from a zero-mean normal distribution with variance σ^2.

$$wing_i = \alpha_j + \beta * length_i + \varepsilon_i$$

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

As this model is written so far, the population effects α_j are estimated as completely unrelated numbers. On the other hand, if we assume that they are exchangeable, we can make the assumption that they are draws from a common distribution with shared (hyper)parameters that are estimated as part of the model fitting. That is, we add to the model the following assumption:

$$\alpha_j \sim Normal(\mu_\alpha, \sigma_\alpha^2)$$

This last assumption is the only difference between a fixed-effects version of the "ANCOVA linear model" and a random-effects version of it! We now use R to fit both models, and plot the resulting regression lines (Figure 3.4). For the random-effects model, we use the REML method, a variant of maximum likelihood that is better suited for mixed models (McCulloch and Searle, 2001; Littell et al., 2008), and fit the model with the function `lmer` in the R package `lme4`, which we load first. Note that in `lmer` (and function `glmer`; see below), terms appearing after the tilde (~) sign are assumed to be fixed effects except when they are between parentheses. For instance, a random intercept term for the levels of the factor `pop` is specified as (1 | pop), and a model with random intercepts *and* random slopes for each population is specified as (body | pop). Although this does not at all become clear from the output, by default in `lmer`, the latter defines a model where the covariance between intercepts and slopes is also an estimated parameter—i.e., where pairs of intercepts and slopes for the populations are treated as draws from a bivariate normal distribution (see p. 161−165 in Kéry, 2010). See below for how to specify the simpler model with two separate univariate normal distributions, one for the intercepts and one for the slopes. Let's now fit the two models first with α_j assumed fixed.

```
summary(lm <- lm(wing ~ pop-1 + body))   # Same as fm2
[ ... ]
Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
popNavarra    6.5296     1.6437    3.973    0.01061 *
popAragon     8.4003     1.5916    5.278    0.00325 **
popCatalonia  8.2630     1.6437    5.027    0.00401 **
body          0.5149     0.1991    2.586    0.04908 *
[ ... ]
```

Next, we fit the model with population effects assumed random. With the function `lmer`, this model is fit in the following parameterization:

$$wing_i = \mu_\alpha + \gamma_j + \beta * length_i + \varepsilon_i$$

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

$$\gamma_j \sim Normal\left(0, \sigma_\gamma^2\right)$$

Here, the mean μ_α is "pulled out" from random effects α_j, and instead random effects γ_j are expressed as zero-mean deviations from the intercept μ_α; hence, $\alpha_j = \mu_\alpha + \gamma_j$.

```
library(lme4)
summary(lmm1 <- lmer(wing ~ (1|pop) + body)) # Fit the model
ranef(lmm1)                                   # Print random effects

Linear mixed model fit by REML ['lmerMod']
Formula: wing ~ (1 | pop) + body
[ ... ]
Random effects:
 Groups   Name        Variance  Std.Dev.
 pop      (Intercept) 0.9861    0.9930     # This is sigma_alpha,
 Residual             0.3023    0.5498     # ... sigma,
Number of obs:  9, groups: pop,  3

Fixed effects:
             Estimate  Std. Error  t  value
(Intercept)   7.7830     1.7034    4.569   # ... mu,
body          0.5084     0.1989    2.556   # ... beta,

ranef(lmm1)                                   # ... and these are the gamma_j.
$pop
          (Intercept)
Navarra    -1.0894238
Aragon      0.6062096
Catalonia   0.4832142
```

We recover $\alpha_j = \mu + \gamma_j$, and compare fixed and random-effects estimates of the population-specific intercepts.

```
alpha_j <- fixef(lmm1)[1]+ranef(lmm1)$pop[,1]
cbind(fixed = coef(lm)[1:3], random = alpha_j)
                    fixed   random
popNavarra   6.529633 6.693583
popAragon    8.400262 8.389216
popCatalonia 8.262966 8.266221
```

To better compare the fixed and random intercept (population) estimates, we plot the lines of best fit under the two versions of the model.

```
par(lwd = 3)
abline(lm$coef[1], lm$coef[4], col = "red", lty = 2)
abline(lm$coef[2], lm$coef[4], col = "blue", lty = 2)
abline(lm$coef[3], lm$coef[4], col = "green", lty = 2)
abline(alpha_j[1], fixef(lmm1)[2], col = "red")
abline(alpha_j[2], fixef(lmm1)[2], col = "blue")
abline(alpha_j[3], fixef(lmm1)[2], col = "green")
abline(fixef(lmm1), col = "black")
legend(6.5, 14, c("Catalonia", "Aragon", "Navarra"), col=c("blue", "green", "red"),
lty = 1, pch = 16, bty = "n", cex = 1.5)
```

In Figure 3.4, you can see how the random-effects regression lines are less extreme than the fixed-effects regression lines, and are pulled in towards their grand mean (i.e., the mean hyperparameter of the assumed distribution for the random effect, represented by the black line); this is shrinkage in action. In a normal-normal random-effects model, the degree of shrinkage, or pulling in towards the grand mean, depends on the ratio between the population variance σ^2_α and the residual variance σ^2. If the population variance σ^2_α is relatively large, we don't see much shrinkage, while if the residual variance σ^2 is relatively large, there is much shrinkage. Fixed-effects estimates can also be described as random-effects estimates with infinite variance hyperparameter σ^2_α.

To fit the random-intercepts model in BUGS, we write the following:

```
for(i in 1:9){       # Data model, loop over the individuals
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha[pop[i]] + beta * body[i]
}
for(j in 1:3){       # Parameter (random effects) model, loop over populations
  alpha[j] ~ dnorm(mu.alpha, tau.alpha) # Mean and precision = 1/variance
}
```

We could also treat the slopes as realizations of a random variable. This would result in the following model (note that now the slope β is indexed by j—i.e., it can vary by population).

$$wing_i = \alpha_j + \beta_j * length_i + \varepsilon_i$$

$$\alpha_j \sim Normal\left(\mu_\alpha, \sigma^2_\alpha\right) \qquad \text{\# Intercepts as random effects}$$

$$\beta_j \sim Normal\left(\mu_\beta, \sigma^2_\beta\right) \qquad \text{\# Slopes as random effects}$$

$$\varepsilon_i \sim Normal\left(0, \sigma^2\right) \qquad \text{\# Same old residual ``random effects''}$$

Another way of describing this model is that we fit three separate normal distributions, of which one has its mean fixed at zero. We can fit this model in R as follows:

```
summary(lmm2 <- lmer(wing ~ body + (1|pop) + (0+body|pop)))
# summary(lmm2 <- lmer(wing ~ body + (body||pop))) # synonym
```

This notation specifies a model with random intercepts (`(1|pop)`) and random slopes (`(0+body|pop)`), but without a correlation between the α_j and the β_j. For our minute data set, this complex model does not make sense, and we don't fit it here (but see Exercise 3).

To fit the random-intercepts model in BUGS, we write the following:

```
for(i in 1:9){    # Data model, loop over the individuals
  wing[i] ~ dnorm(mean[i], tau)
  mean[i] <- alpha[pop[i]] + beta[pop[i]] * body[i]
}
for(j in 1:3){    # Parameter (random effects) model, loop over populations
  alpha[j] ~ dnorm(mu.alpha, tau.alpha) # Model for intercepts
  beta[j] ~ dnorm(mu.beta, tau.beta)    # Model for slopes
}
```

3.4.2 RANDOM EFFECTS FOR A POISSON DATA DISTRIBUTION: POISSON-NORMAL GLMM

Finally, we illustrate a traditional Poisson random-effects, or mixed, model by fitting to the mite counts a Poisson GLMM with random intercepts. This is a Poisson-normal mixed model or an HM, because the distribution of the data is Poisson and the random population effects are assumed to be draws from a normal:

$$mites_i \sim Poisson(\lambda_i)$$

$$\log(\lambda_i) = \alpha_j + \beta * length_i$$

$$\alpha_j \sim Normal(\mu_\alpha, \sigma_\alpha^2)$$

```
summary(glmm <- glmer(mites ~ body + (1|pop), family = poisson))
Generalized linear mixed model fit by maximum likelihood ['glmerMod']
 Family: poisson ( log )
Formula: mites ~ body + (1 | pop)
[ ... ]
Random effects:
 Groups Name        Variance  Std.Dev.
 pop    (Intercept)  0.08535   0.2922
Number of obs: 9, groups: pop, 3

Fixed effects:
             Estimate  Std. Error   z value  Pr(>|z|)
(Intercept)   -5.8718     2.2206    -2.644   0.00819 **
body           0.8351     0.2558     3.265   0.00110 **
```

The code in BUGS looks virtually identical to the algebraic description of the model:

```
for(i in 1:9){     # Data model, loop over the individuals
  mites[i] ~ dpois(lambda[i])
  log(lambda[i]) <- alpha[pop[i]] + beta * body[i]
}
for(j in 1:3){     # Parameter (random effects) model, loop over populations
  alpha[j] ~ dnorm(mu.alpha, tau.alpha) # Mean and precision = 1/variance
}
```

3.5 SUMMARY AND OUTLOOK

This completes our introduction to the main “ingredients” of HMs: linear models, GLMs, and random-effects models. Random effects are simply a consequence of having two or more linked probability models (i.e., an HM) for two or more outcomes from random variables, where the latent (unobserved) outcomes are the random effects. To model structure in the expected outcome of the random effects, we usually specify linear models, typically after applying a link function, exactly as for GLMs. Known structure in the mean of GLMs at each level of an HM can be accommodated with covariates, offsets, random effects, or possibly with more complex modeling of the variance (e.g., distance-related variance-covariance matrices; see Chapters 21 and 22). The ability to fit nonstandard linear models for factors and continuous covariates within the framework of the HMs in this book represents about half of the challenge for you to start to “walk” as a BUGS modeler. Moreover, if you have a good understanding of linear models, GLMs, and random effects, then you are in really good shape for most of the applied statistical analyses that are required of a modern ecologist.

Of course, you may also want to fit nonlinear models. These may be more mechanistic and directly suggested by subject matter knowledge (Pinheiro and Bates, 2000), and they may yield more reliable extrapolations beyond the observed range of the data. Though we do not show this here, nonlinear models may be fit naturally in BUGS (see, e.g., Chapter 19 in Gelman et al., 2014). In addition, generalized additive models (Hastie and Tibshirani, 1990; Wood, 2006) may be fit in BUGS using penalized splines, a sort of mixed model (Crainiceanu et al., 2005; Gimenez et al., 2006a,b; Strebel et al., 2014; see also Section 10.14).

In this chapter, we looked at some of the simplest traditional HMs, or GLMMs: a normal-normal mixed model and a Poisson-normal mixed model. In far more than 90% of applied statistical analyses in ecology, the distribution for the unobserved random variable—i.e., the random-effects distribution—is assumed to be normal. In sharp contrast, in most HMs in this book except for some models in Chapter 12, we will not have such traditional GLMMs, but rather will encounter nonstandard GLMMs with random effects that are assumed to be draws from a Bernoulli or Poisson distribution. This is perhaps surprising at first, and may strike you as strange; but nevertheless, the principle of hierarchical modeling, that a complex model is built as a sequence of simpler models, in the manner of elemental GLMs, is clearly retained for these hierarchical (or “sequential”) models.

In the very simple HMs at the end of this chapter, specification of one or more group effects and a residual (which is implicit in the Poisson case) represents a simple form of modeling of the variance of the response by partitioning variance among hierarchical levels in the model. However, variances may also be modeled more directly—e.g., by known covariates in a linear model for the log-transformed variance. See Sections 6.11.2.2 and 11.6.3 for examples of such covariate modeling of a variance parameter in the context of an N-mixture and a community occupancy model, respectively. We may even add random effects into the linear predictor for variances of a model; see Lee et al. (2006) for examples of such doubly hierarchical GLMMs. Parametric modeling of spatial autocorrelation (see Chapters 21 and 22) could be described as a case where the variance of a response is modeled as a function of a distance covariate.

In Chapter 5, we will fit in BUGS most of the model types introduced in this chapter and compare their estimates with classical maximum likelihood estimates. Thus, Chapter 3 has provided you, in a very applied way, an illustration of the statistical theory for the GLMs and simple mixed models that we will fit with BUGS in Chapter 5. But first, in Chapter 4, we introduce data simulation, which is an extremely important topic for applied statistical analysis (Gelman and Hill, 2007), and assemble a data set to practice BUGS model fitting in Chapter 5.

EXERCISES

1. In Sections 3.2.1 and 3.2.2, fit ANCOVA models with `sex` instead of `pop`, and `color` instead of `body`.
2. Fit some or all of the linear models in Sections 3.2.1–3.2.4 to a Poisson, Bernoulli, and binomial response—i.e., to the mite counts, the indicator for heavy parasitization, and the damage variable (as per Sections 3.3.1–3.3.3).
3. See whether you can fit a model with random intercepts and random slopes (as in Section 3.4.1) to a clone of the toy data set that is 10 times larger. Here is that data set:

```
# Define and plot data (10 times larger data set than the toy data set)
clone.size <- 10          # clone size
pop <- factor(rep(c(rep("Navarra", 3), rep("Aragon", 3), rep("Catalonia", 3)),
levels = c("Navarra", "Aragon", "Catalonia"), clone.size))
wing <- rep(c(10.5, 10.6, 11.0, 12.1, 11.7, 13.5, 11.4, 13.0, 12.9), clone.size)
body <- rep(c(6.8, 8.3, 9.2, 6.9, 7.7, 8.9, 6.9, 8.2, 9.2), clone.size)
sex <- rep(factor(c("M","F","M","F","M","F","M","F","M"), levels = c("M", "F")),
clone.size)
mites <- rep(c(0, 3, 2, 1, 0, 7, 0, 9, 6), clone.size)
color <- rep(c(0.45, 0.47, 0.54, 0.42, 0.54, 0.46, 0.49, 0.42, 0.57), clone.size)
damage <- rep(c(0,2,0,0,4,2,1,0,1), clone.size)
```

4. Fit the analogous Poisson GLMM to the mite counts for the larger clone of the data set.
5. Fit a random-intercepts and a random-intercepts-and-slopes model (as in Section 3.4.2) for a binomial response (for `damage`).