CHAPTER

6

Key Components of (Generalized) Linear Models: Statistical Distributions and the Linear Predictor

OUTLINE

Introduction to WinBUGS for Ecologists
DOI: 10.1016/B978-0-12-378605-0.00006-5

6.1 INTRODUCTION

All models in this book explain the variation in an observed response as being composed of a deterministic and a stochastic part. Thus, the concept of these models is this:

response = deterministic part + stochastic part

The deterministic part is also called the systematic part and the stochastic the random part of a model. It is the presence of the stochastic part that makes a model a *statistical*, rather than simply a mathematical model. For the description of the stochastic part of the model, we use a statistical distribution. In order to choose the "right" distribution, we need to know the typical sampling situation that leads to one rather than another distribution. Sampling situation means how the studied objects were chosen and how and what characteristic was measured. The quantity measured or otherwise assessed, the variation of which one wants to explain, is the response. In this chapter, we briefly review four of the most common statistical distributions that are used to capture the variability in a response: the normal, the uniform, the binomial, and the Poisson. These are virtually all the distributions that you need to know for this book. In addition, they are also those that underlie a vast range of statistical techniques that ecologists commonly employ.

Linear models are so called because they assume that the mean (i.e., the expected) response can be treated as the result of explanatory variables whose effects add together. Effects are additive regardless of whether the explanatory variables are continuous (covariates, regressors) or discrete (factors). To be able to specify how exactly we think the explanatory variables affect the response, we need to understand the so-called linear predictor of the model, the design matrix, and different parameterizations of a linear model. It is the design matrix along with the observed values of the covariates that makes up the deterministic part of a linear statistical model. They combine to form the linear predictor, i.e., the expected response, which is the value of the response that we would expect to observe in the absence of a stochastic component in the studied system.

In programs such as R or GenStat, the specification of the design matrix, and hence the linear predictor, just "happens" internally without us having to know how exactly this works. All we type is a formula to describe the linear model as in `response ~ effect1 + effect2`. In contrast and unfortunately (or actually, fortunately!), using WinBUGS requires us to know exactly what kind of linear model we want to fit because we have to describe this model at a very elementary level. Therefore, in this chapter, we review some of the basics of linear models. I expect them to be useful also for your general understanding of linear statistical models,

whether analyzed in a Bayesian or frequentist mode of inference. We deal first with distributions (the stochastic part of a model) and then with the design matrix and the linear predictor (the deterministic part).

6.2 STOCHASTIC PART OF LINEAR MODELS: STATISTICAL DISTRIBUTIONS

Parametric statistical modeling means describing a caricature of the "machine" that plausibly could have produced the numbers we observe. The machine is nature, and *nature is stochastic*, i.e., nature is never fully predictable. An element of chance in the observed outcome of something, e.g., body mass, does not mean that the outcome has no reason for happening. There is always a reason; we simply don't know it. Chance just means that there are a few or many unrecognized factors that affect the outcome, but we either don't know or haven't measured them, or that we don't fully understand the relationship between these factors and an outcome. Things whose outcome is affected by chance and thus are only predictable up to a certain degree are called *random variables*. At some level, almost anything in nature is best thought of as a random variable.

However, random variables are seldom totally unpredictable; instead, the *combined effect* of all the unmeasured factors can often reasonably well be described by a mathematical abstraction: a probability distribution. A probability distribution assigns to each possible realization (value or event) of a random variable a probability of occurrence. To be a proper probability distribution, that description must be complete, i.e., all possible realizations must be described so that the sum of their probabilities of occurring is equal to 1.

The types of events vary greatly and may be binary like a coin flip (heads/tail), a survival event (dead/survived), or sex (male/female). They may be categorical like hair color (brown/blonde/red/white/gray), nationality (Swiss/French/Italian), or geographical location. (Note that a binary random variable is just a special case of a categorical one.) They may be counts, like the number of birds in a sampling quadrat, the number of people in a park or pub, or the number of boys in a school class (Note the slight difference in this last kind of count?). Or they can be measurements, like body mass, wing length, or diameter of a tree.

Probability distributions are themselves governed (described) by one or a few parameters, and their actual form depends on the particular values of these parameters. A probability distribution can be described using a formula, a table, or a picture. Statisticians usually prefer formulae and ecologists pictures. The advantage of a formula is that it is completely general, while a picture can only ever show how a distribution looks like for particular parameter values. Depending on these values, different

distributions can yield very similar pictures or same distributions can look very different. Nevertheless, there are some typical features of many distributions, and it may be worthwhile to describe these, so you have a better chance of recognizing a distribution when you meet it.

Of the four distributions we encounter in this book, two are discrete and have non-negative values, i.e., they can only take on integer values from 0, 1, and upwards. They are the binomial distribution and the Poisson distribution. Then, we have two continuous distributions, i.e., that can take on any value, within measuring accuracy. One is the famous normal distribution, which is defined on the entire real line (from $-\infty$ to ∞), and the other is the uniform distribution, which is usually defined between its lower and upper limits.

Sometimes, people are astonished at how modelers know how to choose the right kind of distribution for describing their random variables. It is true that there is some arbitrary element involved and, often, there is more than a single distribution that could be used to describe the output of the machine that produced our observations. However, very frequently the circumstances of the data and how these were collected quite firmly point to one rather than another distribution as the most appropriate description of the number-generating machine and therefore also of the random variability in the output of that machine, the response.

Here, I describe circumstances in which each distribution would arise during sampling and then provide a pictorial example for each. I also give few lines of R code that are required to create random samples from each distribution and then plot them in a histogram for checking how selected parameter values influence the shape of a distribution. You can play around with different parameter values and so get a feel for how the distributions change as parameters are varied.

Among the four distributions (normal, uniform, Poisson, and binomial), one big divide is whether your response (i.e., the data) is discrete or continuous. Counts are discrete and point to the two latter distributions, while measurements are continuous and point to the two former distributions. In practice, there is some overlap. Since measurement accuracy is finite, every continuous random variable is recorded in a discrete way. However, this is usually of no consequence. On the other hand, under certain circumstances (e.g., large sample sizes, many observed unique values), the two discrete distributions can often be well approximated by a normal distribution. For instance, large counts in practice are often modeled as coming from a normal distribution.

What follows are brief vignettes of the four distributions we use throughout this book. Further features of these and many other distributions for selected parameter values can be studied in R (type `?rnorm`, `?runif`, `?rpois`, or `?rbinom` to find out more).

6.2.1 Normal Distribution

Sampling situation: Measurements are taken, which are affected by a large number of effects that act in an additive way.
Classical examples: They include body size and other linear measurements. In WinBUGS, they are also used to specify ignorance in priors when the precision is small (meaning the variance large).
Varieties: When effects are multiplicative instead of additive, we get the log-normal distribution. A log-normal random variable can be transformed into a normal one by log-transforming it.
Typical picture: It is the Gaussian bell curve, i.e., symmetrical, single hump, more or less long tails. In small samples, it can look remarkably irregular, e.g., skewed.
Mathematical description: It includes two parameters, the mean (location) and standard deviation (spread, average deviation from the mean) or, equivalently, the variance (squared standard deviation). In WinBUGS, spread is specified as precision = 1/variance.
Specification in WinBUGS:

```
x ~ dnorm(mean, tau) # note tau = 1/variance
```

R code to draw n *random number with specified parameter(s) and plot a histogram (see Fig. 6.1)*:

```
n <- 100000                    # Sample size
mu <- mean <- 600              # Body mass of male peregrines
sd <- st.dev <- 30             # SD of body mass of male peregrines
```

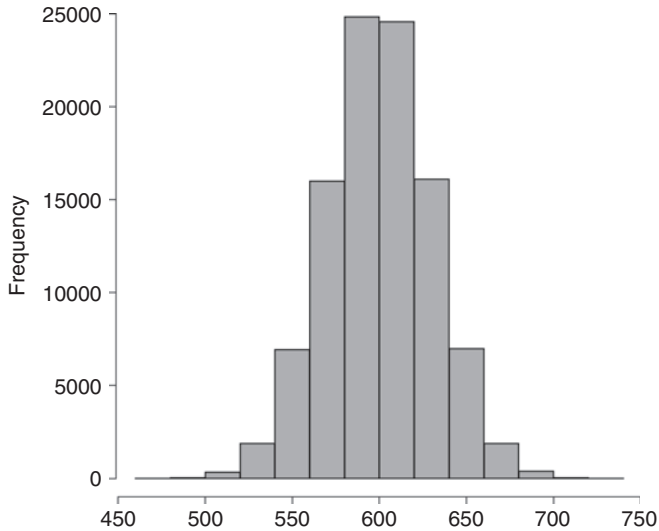

FIGURE 6.1 Sample histogram of normal distribution.

```
sample < rnorm(n = n, mean = mu, sd = sd)
print(sample, dig = 4)
hist(sample, col = "grey")
```

Mean and standard deviation:

$$E(y) = \mu \quad \text{mean}$$
$$\text{sd}(y) = \sigma \quad \text{sd}$$

6.2.2 Continuous Uniform Distribution

Sampling situation: Measurements are taken, which are all equally likely to occur in a certain range of values.
Classical examples: In WinBUGS, this distribution is typically used to specify ignorance in a prior (as an alternative to a "flat" normal).
Varieties: The distribution can be discrete uniform, e.g., the result of rolling a die with the response being anything from 1 to 6.
Typical picture: It is a rectangle shape (with variation due to sampling variability).
Mathematical description: It includes two parameters, lower (a) and upper limits (b).
Specification in WinBUGS:

```
x ~ dunif(lower, upper)
```

R code to draw n *random number with specified parameter(s) and plot histogram (see Fig. 6.2)*:

```
n < 100000              # Sample size
a < lower.limit < 0     #
b < upper.limit < 10    #

sample < runif(n = n, min = a, max = b)
print(sample, dig = 3)
hist(sample, col = "grey")
```

Mean and standard deviation:

$$E(y) = (a+b)/2 \quad \text{mean}$$
$$sd(y) = \sqrt{(b-a)^2/12} \quad \text{sd}$$

6.2.3 Binomial Distribution: The "Coin-Flip Distribution"

Sampling situation: When N available things all have the same probability p of being in a certain state (e.g., being counted, or having a certain attribute, like being male or dead), then the number x that is actually counted in that sample, or has that attribute, is binomially distributed.

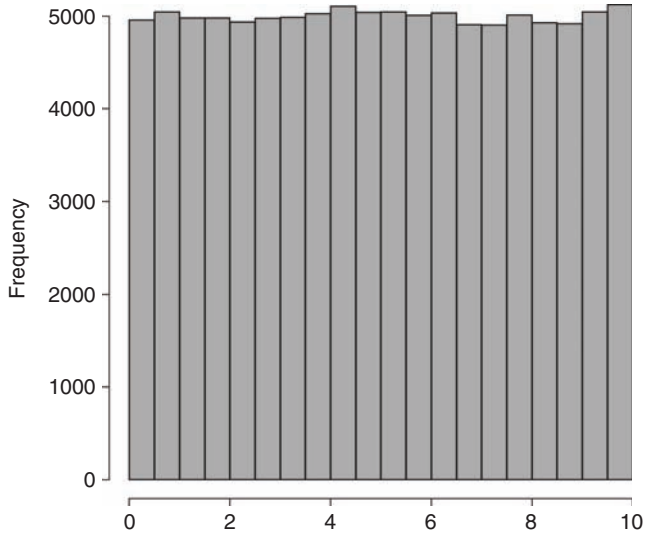

FIGURE 6.2 Sample histogram of uniform distribution.

Classical examples: Number of males in a clutch, school class, or herd of size N; number of times heads shows up among N flips of a coin; number of times you get a six among $N = 10$ rolls of a die; and the number of animals among those N present that you actually detect.
Varieties: The Bernoulli distribution corresponds to a single coin flip and has only a single parameter, p. Actually, a binomial is the sum of N Bernoullis (or coin flips).
Typical picture: It varies a lot but strictly speaking always discrete. Normally, it is skewed, but skewness depends on the actual values of the parameters. The binomial distribution is symmetrical for $p = 0.5$.
Mathematical description: It includes one or two parameters, the probability of being chosen or having a certain trait (male, dead), often called success probability p, and the "binomial total" N, which is the sample or trial "size." N represents a ceiling to a binomial count; this is an important distinction to the similar Poisson distribution. Usually, N is observed and therefore not a parameter (but see Chapter 21). Aim of modeling is typically estimation and modeling of p (but sometimes also of N; see Chapter 21).
Important feature: The binomial comes with a "built-in" variance equal to $N * p * (1 - p)$, i.e., the variance is a function of the mean, which is $N * p$. See also Fig. 6.3.
Specification in WinBUGS:

```
x ~ dbin(p, N) # Note order of parameters !
```

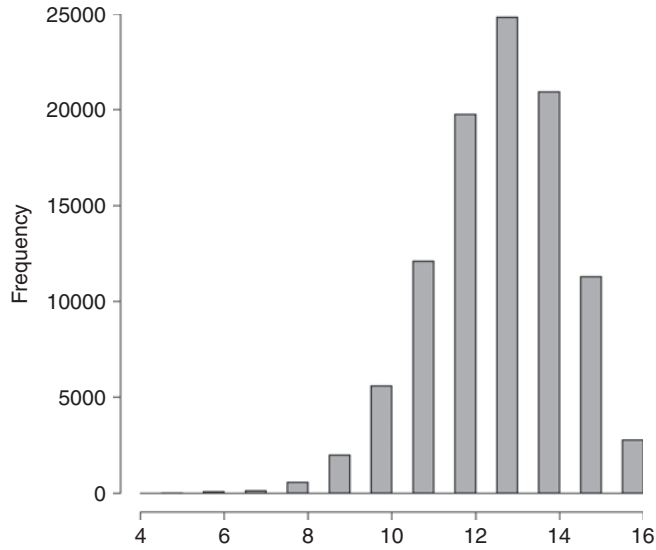

FIGURE 6.3 Sample histogram of binomial distribution with $N=16$ and $p=0.8$.

Bernoulli:

```
x ~ dbern(p)
```

R code to draw n *random number with specified parameter(s) and plot histogram (Fig. 6.3)*:

```
n <- 100000          # Sample size
N <- 16              # Number of individuals that flip the coin
p <- 0.8             # Probability of being counted (seen), dead or a male

sample <- rbinom(n = n, size = N, prob = p)
print(sample, dig = 3)
hist(sample, col = "grey")
```

Extra comment: This picture can be thought to give the frequency with which we count 4, 5, 6, …, 16 greenfinches in a monitoring plot that has a population of $N = 16$ individuals and where each greenfinch has a probability of being seen (= counted) of exactly $p = 0.8$. It is important, though not sufficiently widely recognized among ornithologists, to note that there is variation in bird counts even under constant conditions, i.e., when N and p are constant (Kéry and Schmidt, 2008).

Mean and standard deviation:

$$E(y) = N * p \qquad \text{mean}$$
$$sd(y) = \sqrt{N * p * (1-p)} \qquad \text{sd}$$

Mean and standard deviation for the Bernoulli are p and $\sqrt{p * (1-p)}$, respectively.

6.2.4 Poisson Distribution

Sampling situation: When things (e.g., birds, cars, or erythrocytes) are randomly distributed in one or two (or more) dimensions and we randomly place a "counting window" along that dimension or in that space and record the number of things, then that number is Poisson distributed.
Classical examples: Number of passing cars during 10-min counts at a street corner; number of birds that fly by you at a migration site, annual number of Prussian soldiers kicked to death by a horse; number of car accidents per day, month, or year, and number of birds or hares per sample quadrat.
Varieties: None really. But the Poisson is an approximation to the binomial when N is large and p is small and can itself be approximated by the normal when the average count, λ, is large, e.g., greater than 10. The negative binomial distribution is an overdispersed version of the Poisson and can be derived by assuming that the Poisson mean, λ, is itself a random variable with another distribution, the gamma. Therefore, it is also referred to as a Poisson–gamma mixture.
Typical picture: It varies a lot but strictly speaking always discrete. It is skewed normally, but skewness depends on the value of lambda.
Mathematical description: It includes a single parameter called λ, which is equal to the mean (= expectation, average count, intensity), as well as the variance (i.e., variance = mean). That is, as for the binomial distribution, the variance is not a free parameter but is a function of the mean.
Important feature: As for the binomial, values from a Poisson distribution are non-negative integers that come with a built-in variance.
Specification in WinBUGS:

```
x ~ dpois(lambda)
```

R code to draw n *random number with specified parameter(s) and plot histogram (see Fig. 6.4)*:

```
n <- 100000          # Sample size
lambda <- 5          # Average # individuals per sample, density
sample <- rpois(n = n, lambda = lambda)
print(sample, dig = 3)

par(mfrow = c(2,1))
hist(sample, col = "grey", main = "Default histogram")
plot(table(sample), main = "A better graph", lwd = 3, ylab = "Frequency")
```

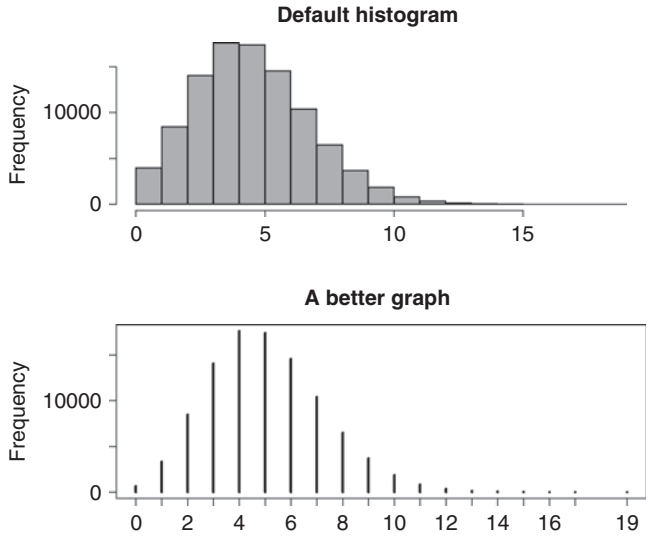

FIGURE 6.4 Sample histogram of Poisson distribution. The lower graph is better since it shows the discrete nature of a Poisson random variable.

Mean and standard deviation:

$$E(y) = \lambda \qquad \text{mean}$$
$$sd(y) = \sqrt{\lambda} \qquad \text{sd}$$

6.3 DETERMINISTIC PART OF LINEAR MODELS: LINEAR PREDICTOR AND DESIGN MATRICES

Linear models describe the expected response as a linear combination of the effects of discrete or continuous explanatory variables. That is, they directly specify the relationship between the response and one or more explanatory variables. I note, in passing, that it is this linear (additive) relationship that makes a statistical model linear, not that a picture of it is a straight line. Most statistically linear models describe relationships that are represented by curvilinear graphs of some kind. Thus, one must differentiate between linear statistical models and linear models that also imply straight-line relationships.

Most ecologists have become accustomed to using a simple formula language when specifying linear models in statistical software. For instance, to specify a two-way analysis of variance (ANOVA) with interaction in R, we just type `y ~ A*B` as an argument of the functions `lm()` or `glm()`. Many of us don't know exactly what typing `y ~ A*B` causes R to do internally, and this may not be fatal. However, to fit a linear model in

WinBUGS, you must know exactly what this and other model descriptions mean and how they can be adapted to match your hypotheses! Therefore, I next present a short guide to the so-called design matrix of a linear or generalized linear model (GLM). We will see how the same linear model can be described in different ways; the different forms of the associated design matrices are called *parameterizations* of a model.

The material in the rest of this chapter may be seen by some as a nuisance. It may look difficult at first and indeed may not be totally necessary when fitting linear models using one of the widely known stats packages. For better or worse, in order to use WinBUGS, you must understand this material. However, understanding design matrices will greatly increase your grasp of statistical models in general. In particular, you will greatly benefit from this understanding when you use other software to conduct more specialist analyses that ecologists typically conduct. For instance, you need to understand the design matrix when you use program MARK (White and Burnham, 1999) to fit any of a very large range of capture–recapture types of models (for instance, most of those described by Williams et al., 2002). Thus, time invested to understand this material is time well spent for most ecologists!

For each element of the response vector, the design matrix *n* index indicates which effect is present for categorical (= discrete) explanatory variables and what "amount" of an effect is present in the case of continuous explanatory variables. The design matrix contains as many columns as the fitted model has parameters, and when matrix-multiplied with the parameter vector, it yields the linear predictor, another vector. The linear predictor contains the expected value of the response (on the link scale), given the values of all explanatory variables in the model. Expected value means the response that would be observed when all random variation is averaged out. For a fuller understanding of some of this GLM jargon, you may need to jump to Chapter 13 (and back again). However, this is not required for an understanding of this chapter.

In the remainder of this chapter, we look at a progression of typical linear models (e.g., t-test, simple linear regression, one-way ANOVA, and analysis of covariance (ANCOVA)). We see how to specify them in R and then find out what R does internally when we do this. We look at how to write these models algebraically and what system of equations they imply. We see different ways of writing what is essentially the same model (i.e., different parameterizations of a model) and how these affect the interpretation of the model parameters. Only understanding all the above allows us to specify these linear models in WinBUGS (though, notably, none of this has anything to do with Bayesian statistics).

For most nonstatisticians, it is usually much easier to understand something with the aid of a numerical example. So to study the design matrices for these linear models, we introduce a toy data set consisting of just six

TABLE 6.1 Our Toy Data Set for Six Snakes

mass	pop	region	hab	svl
6	1	1	1	40
8	1	1	2	45
5	2	1	3	39
7	2	1	1	50
9	3	2	2	52
11	3	2	3	57

data points (Table 6.1). Let's imagine that we measured body mass (`mass`, in units of 10 g; a continuous response variable) of six snakes in three populations (`pop`), two regions (`region`), and three habitat types (`hab`). These three discrete explanatory variables are factors with 3, 2, and 3 levels, respectively. We also measured a continuous explanatory variable for each snake, snout–vent length (`svl`).

Here is the R code for setting up these variables:

```
mass < c(6, 8, 5, 7, 9, 11)
pop < factor(c(1,1,2,2,3,3))
region < factor(c(1,1,1,1,2,2))
hab < factor(c(1,2,3,1,2,3))
svl < c(40, 45, 39, 50, 52, 57)
```

We use `factor()` to tell R that the numbers in this variable are just names that lack any quantitative meaning. Next, we use the R functions `lm()` or `glm()` to specify different linear models for these data and inspect what this exactly means in terms of the design matrix.

6.3.1 The Model of the Mean

What if we just wanted to fit a common mean to the mass of all six snakes? That is, we want to fit a linear model with an intercept only (see Chapters 4 and 5). In R, this is done simply by issuing the command:

```
lm(mass ~ 1)
```

The 1 implies a covariate with a single value of one for every snake. One way to write this model algebraically is this:

$$\text{mass}_i = \mu + \varepsilon_i$$

That is, we imagine that the mass measured for snake i is composed of an overall mean, μ, plus an individual deviation from that mean

called ε_i. Of course, the latter is the residual for snake i. If we want to estimate the mean within a linear statistical model as above, then we also need an assumption about these residuals, for instance $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$. That is, we assume that the residuals are normally distributed around μ with a variance of σ^2.

Let's look at the design matrix of that model using the function `model.matrix()`. We see that it consists just of a vector of ones and that this vector is termed the intercept by R.

```
model.matrix(mass ~ 1)
```

In the rest of this book, we make extensive use of this very useful R function, `model.matrix()`. Next, we consider in more detail some less trivial linear statistical models.

6.3.2 t-Test

When we are interested in the effect of a single, binary explanatory variable such as `region` on a continuous response such as `mass`, we can conduct a t-test (see Chapter 7). Here is the specification of a t-test as a linear model in R:

```
lm(mass ~ region)
```

Some might wonder now: "But where is the p-value?" However, the most important thing in the linear model underlying a t-test is not the p-value but the vector of parameter estimates. R is a respectable statistics program and thus presents the most important things first. But of course, there are several ways the p-value might be obtained, for instance, by `summary(lm(mass ~ region))`.

In the methods section of a research paper, you would not describe your model with the R description above; rather, you could describe it algebraically. Of course, for such a simple model as that underlying the t-test, you would simply mention that you used a t-test, but for the sake of getting some mental exercise, let's do it now in a more general way. One way of describing the linear model that relates snake mass to region is with the following statement:

$$\text{mass}_i = \alpha + \beta * \text{region}_i + \varepsilon_i$$

This means that the mass of snake i is made up of the sum of three components: a constant α, the product of another constant (β) with the value of the indicator for the region in which snake i was caught, and a third term ε_i that is specific to each snake i. The last term, ε_i, is the residual, and to make this mathematical model a statistical model, we need some assumption about how these residuals vary among individuals. One of the

simplest assumptions about these unexplained, snake-specific contributions to mass is that they follow a normal distribution centered on zero and with a spread (standard deviation or variance) that we are going to estimate. We can write this as follows:

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

This way of writing the linear model underlying a t-test immediately clarifies a widespread confusion about what exactly needs to be normally distributed in normal linear models: it is the residuals and not the raw response. Thus, if you are concerned about the normality or not of a response variable, you must first fit a linear model and only then inspect the residual distribution.

A second and equivalent way to write the linear model for the t-test algebraically is this:

$$\text{mass}_i \sim \text{Normal}(\alpha + \beta * \text{region}_i, \sigma^2)$$

This way to write down the model resembles more the way in which we specify it in WinBUGS, as we will see. It clarifies that the mean response is not the same for each snake. Moreover, in this way to write the model, $\alpha + \beta * \text{region}_i$ represents the deterministic part of the model and $\text{Normal}(..., \sigma^2)$ its stochastic part.

Finally, we might also write this model like this: $\text{mass}_i = \mu_i + \varepsilon_i$, which again says that the mass of snake i is the sum of a systematic effect, embodied by the mean or expected mass μ_i, and a random effect ε_i. The expected mass of snake i, μ_i, consists of $\alpha + \beta * \text{region}_i$. This is called the linear predictor of the model, and the residuals are assumed to follow a normal distribution centered on the value of the linear predictor, as before.

Regardless of how we describe the model, we must know what the two-level variable named `region` actually looks like in the analysis of the linear model underlying the t-test. When we define `region` to be a factor and then fit its effect, what R does internally is to convert it into a design matrix with two columns, one containing only ones (corresponding to the intercept) and the other being a single indicator or dummy variable. That is a variable containing just ones and zeroes that indicates which snake was caught in the region indicated in R's name for that column in the design matrix.

We can look at this using the function `model.matrix()`:

```
model.matrix(~region)
```

```
> model.matrix(~region)
 (Intercept)   region2
1           1         0
2           1         0
3           1         0
```

```
4          1          0
5          1          1
6          1          1
[...]
```

(This is what the R default *treatment contrasts* yield; type `?model.matrix` to find out what other contrasts are possible in R.) Therefore, the indicator variable named `region2` contains a one for those snakes in region 2.

Fitting the model underlying the t-test to our six data points with `region` defined in this way implies a system of equations, and it is very instructive to write it down:

$$\begin{aligned}
6 &= \alpha * 1 + \beta * 0 + \varepsilon_1 \\
8 &= \alpha * 1 + \beta * 0 + \varepsilon_2 \\
5 &= \alpha * 1 + \beta * 0 + \varepsilon_3 \\
7 &= \alpha * 1 + \beta * 0 + \varepsilon_4 \\
9 &= \alpha * 1 + \beta * 1 + \varepsilon_5 \\
11 &= \alpha * 1 + \beta * 1 + \varepsilon_6
\end{aligned}$$

Here, one sees why the intercept is often represented by a 1: it is always present and identical for all the values of the response variable. To get a solution for this system of equations, i.e., to obtain values for the unknowns α and β that are "good" in some way, we need to define some criterion for dealing with the residuals ε_i. Usually, in this system of equations, the unknowns α and β are chosen such that the sum of the squared residuals is minimal. This is called the least-squares method, and for normal GLMs (again, see Chapter 13), the resulting parameter estimates for α and β are equivalent to those obtained using the more general maximum likelihood method.

A more concise way of writing the same system of equations is using vectors and matrices:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

We see again that the response is made up of the value of the linear predictor plus the vector of residuals. The linear predictor consists of the product of the *design matrix* (also called model matrix or X matrix) and the *parameter vector*. In a sense, the design matrix contains the "weights" with which α and β enter the linear predictor for each snake i. The linear predictor, i.e., the result of that matrix multiplication, is another

vector containing the expected value of the response, given the covariates, for each snake i.

What is the interpretation of these parameters? This is a crucial topic and one that depends entirely on the way in which the design matrix is specified for a model. It is clear from looking at the extended version of the system of equations above (or the matrix version as well) that α must represent the expected mass of a snake in region 1 and β is the difference in the expected mass of a snake in region 2 compared with that of a snake in region 1. Thus, region 1 serves as a baseline or reference level and in the model becomes represented by the intercept parameter, and the *effect* of region 2 is parameterized as a difference from that baseline. Let's look again at an analysis of this model parameterization in R:

```
lm(mass ~ region)
> lm(mass ~ region)

Call:
lm(formula = mass ~ region)

Coefficients:
(Intercept)    region2
        6.5        3.5
```

We recognize the value of the intercept as the mean mass of snakes in region 1 and the parameter called `region2` as the difference between the mass in region 2 and that in region 1, i.e., the effect of region 2, or here the effect of region. Hence, we may call this an *effects parameterization* of the t-test model. Actually, in the output, R labels the parameter for region 2 by the name of its column in the design matrix and that may be slightly misleading.

Importantly, this is not the only way in which a linear model representing a t-test for a difference between the two regions might be specified. Another way how to set up the equations, which is equivalent to the one above and is called a *reparameterization* of that model, is this:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

What do the parameters α and β mean now? The interpretation of α has not changed; it is the expected mass of a snake in region 1. However, the interpretation of parameter β is different, since now it is the expected body mass of a snake in region 2. Thus, in this parameterization of the linear model underlying the t-test, the parameters directly represent the group

means, and we may call this a *means parameterization* of the model. This model can be fitted in R by removing the intercept:

```
model.matrix(~region 1)
> model.matrix(~region 1)
   region1  region2
1        1        0
2        1        0
3        1        0
4        1        0
5        0        1
6        0        1
[ ... ]

lm(mass~region 1)
> lm(mass~region 1)

Call:
lm(formula = mass ~ region  1)

Coefficients:
region1    region2
    6.5       10.0
```

Why would we want to use these different parameterizations of the same model? The answer is that they serve different aims: the effects parameterization is more useful for testing for a difference between the means in the two regions; this is equivalent to testing whether the effect of region 2, i.e., parameter b, is equal to zero. In contrast, for a summary of the analysis, we might prefer the means parameterization and directly report the estimated expected mass of snakes for each region. However, the two models are equivalent; for instance, the sum of the two effects in the former parameterization is equal to the value of the second parameter in the latter parameterization, i.e., 6.5 + 3.5 = 10.

6.3.3 Simple Linear Regression

To examine the relationship between a continuous response (`mass`) and a continuous explanatory variable such as `svl`, we would specify a simple linear regression in R (see also Chapter 8):

```
lm(mass ~ svl)
```

This model is written algebraically in the same way as that underlying the t-test:

$$\text{mass}_i = \alpha + \beta * \text{svl}_i + \varepsilon_i$$

and

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

Or like this:

$$\text{mass}_i \sim \text{Normal}(\alpha + \beta * \text{svl}_i, \sigma^2)$$

The only difference to the t-test lies in the contents of the explanatory variable, `svl`, which may contain any real number rather than just the two values of 0 and 1. Here, `svl` contains the lengths for each of the six snakes. We inspect the design matrix that R builds on issuing the above call to `lm()`:

```
model.matrix(~svl)
> model.matrix(~svl)
  (Intercept)  svl
1           1   40
2           1   45
3           1   39
4           1   50
5           1   52
6           1   57
[]
```

Thus, fitting this simple linear regression model to our six data points implies solving the following system of equations:

$$\begin{aligned}
6 &= \alpha + \beta * 40 + \varepsilon_1 \\
8 &= \alpha + \beta * 45 + \varepsilon_2 \\
5 &= \alpha + \beta * 39 + \varepsilon_3 \\
7 &= \alpha + \beta * 50 + \varepsilon_4 \\
9 &= \alpha + \beta * 52 + \varepsilon_5 \\
11 &= \alpha + \beta * 57 + \varepsilon_6
\end{aligned}$$

Again, a more concise way of writing the same equations is by using vectors and a matrix:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 40 \\ 1 & 45 \\ 1 & 39 \\ 1 & 50 \\ 1 & 52 \\ 1 & 57 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

The design matrix contains an intercept and another column with the values of the covariate `svl`. The interpretation of the parameters α and β is thus that of a baseline, representing the expected value of the

response (mass) at a covariate value of svl = 0, and a difference or an effect, representing the change in mass for each unit change in svl. Equivalently, this effect is the slope of the regression of mass on svl or the effect of svl.

Let's look at an analysis of this model in R:

```
lm(mass~svl)
> lm(mass~svl)

Call:
lm(formula = mass ~ svl)

Coefficients:
(Intercept)       svl
     5.5588    0.2804
```

The intercept is biologically nonsense; it says that a snake of zero length weighs –5.6 mass units! This illustrates that a linear model can only ever be a useful characterization of a biological relationship over a restricted range of the explanatory variables. To give the intercept a more sensible interpretation, the model could be reparameterized by transforming svl to svl mean(svl). Fitting this centered version of svl will cause the intercept to become the expected mass of a snake at the average of the observed size distribution.

Again, this is not the only way to specify a linear regression of mass on svl, and we could again remove the intercept as in the t-test. However, removing the intercept is not just a reparameterization of the same model; instead, it changes the model and forces the regression line to go through the origin. The design matrix then contains just the values of the covariate svl:

```
model.matrix(~svl 1)
> model.matrix(~svl 1)
  svl
1   40
2   45
3   39
4   50
5   52
6   57
[ ... ]

lm(mass~svl 1)
> lm(mass~svl 1)

Call:
lm(formula = mass ~ svl  1)

Coefficients:
   svl
0.1647
```

We see that the estimated slope is less than before, which makes sense, since previously the estimate of the intercept was negative and now it is forced to be zero. In most instances, the no-intercept model is not very sensible, and we should have strong reasons for forcing the intercept of a linear regression to be zero. Also, the lower limit of the observed range of the covariate should be close to, or include, zero.

6.3.4 One-Way Analysis of Variance

To examine the relationship between `mass` and `pop`, a factor with three levels, we would specify a one-way ANOVA in R:

```
lm(mass ~ pop)
```

There are different equivalent ways in which to write this model algebraically; see also Chapter 9. Here, we focus on the effects and the means parameterizations and show how their design matrices look. For the mass of individual i in population j, one way to write the model is like this:

$$\text{mass}_i = \alpha + \beta_{j(i)} * \text{pop}_i + \varepsilon_i$$

and

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

Another way to specify the same model is this:

$$\text{mass}_i \sim \text{Normal}(\alpha + \beta_{j(i)} * \text{pop}_i, \sigma^2)$$

This parameterization means to set up the design matrix in an *effects* format, i.e., with an intercept α, representing the mean for population 1, plus indicators for every population except the first one, representing the differences between the means in these populations and the means in the reference population. (The choice of reference is arbitrary and has no influence on inference.). For n populations, there are $n - 1$ β_j parameters, which are indexed by factor level j.

An algebraic description of the *means* parameterization of the one-way ANOVA would be this:

$$y_i = \alpha_{j(i)} * \text{pop}_i + \varepsilon_i$$

and

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

Here, the design matrix is set up to indicate each individual population, and the parameters α_j represent the mean body mass of snakes in each population j.

We look at the two design matrices associated with the two parameterizations.

First, the effects parameterization, which is the default in R:

```
model.matrix(~pop)
```

```
> model.matrix(~pop)
  (Intercept)  pop2  pop3
1           1     0     0
2           1     0     0
3           1     1     0
4           1     1     0
5           1     0     1
6           1     0     1
[ ... ]
```

Second, the means parameterization, which is specified in R by removing the intercept:

```
model.matrix(~pop 1)
> model.matrix(~pop 1)
   pop1  pop2  pop3
1     1     0     0
2     1     0     0
3     0     1     0
4     0     1     0
5     0     0     1
6     0     0     1
[ ... ]
```

Again, the interpretation of the first parameter of the model is the same for both parameterizations: it is the mean body mass in population 1. However, while in the effects parameterization, the parameters 2 and 3 correspond to differences in the means (above), and they represent the actual expected body mass for snakes in each of the three populations in the means parameterization (below). Also note how the first parameter changes names when going from the former to the latter parameterization of the one-way ANOVA.

Here is the matrix–vector description of the effects ANOVA model applied to our toy snake data set:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta_2 \\ \beta_3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

And here is the means parameterization:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Again, we see clearly that the interpretation of the first parameter in the model is not affected by the parameterization chosen, but those for the second and third are.

Let's fit the two versions of the ANOVA in R, first the effects and then the means model:

```
lm(mass~pop)                # Effects parameterization (R default)
> lm(mass~pop)

Call:
lm(formula = mass ~ pop)

Coefficients:
(Intercept)         pop2         pop3
          7            1            3
lm(mass~pop 1)              # Means parameterization
> lm(mass~pop 1)

Call:
lm(formula = mass ~ pop  1)

Coefficients:
pop1  pop2  pop3
   7     6    10
```

Each parameterization is better suited to a different aim: the effects model is better for testing for differences and the means model is better for presentation.

6.3.5 Two-Way Analysis of Variance

A two-way or two-factor ANOVA serves to examine the relationship between a continuous response, such as `mass`, and two discrete explanatory variables, such as population (`region`) and `habitat (hab)`, in our example (see also Chapter 10). Importantly, there are two different ways in which to combine the effects of two explanatory variables: additive (also called main effects) and multiplicative (also called interaction effects). In addition, there is the possibility to specify these models using an effects parameterization or a means parameterization. We consider each one in turn.

To specify a main-effects ANOVA with `region` and `hab` in R, we would write

```
lm(mass ~ region + hab)
> lm(mass ~ region + hab)

Call:
lm(formula = mass ~ region + hab)

Coefficients:
(Intercept)     region2     hab2     hab3
       6.50         3.50     0.25     0.25
```

To avoid overparameterization (i.e., trying to estimate more parameters than the available data allow us to do), we need to arbitrarily set to zero the effects of one level for each factor. The effects of the remaining levels then get the interpretation of differences relative to the base level. It does not matter which level is used as a baseline or reference, but often stats programs use the first or the last level of each factor. R sets the effects of the first level to zero.

In our snake toy data set, for the mass of individual i in region j and habitat k, we can write this model as follows:

$$\text{mass}_i = \alpha + \beta_{j(i)} * \text{region}_i + \delta_{k(i)} * \text{hab}_i + \varepsilon_i$$

and

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

Here, α is the expected mass of a snake in habitat 1 and region 1. There is only one parameter β_j, so the subscript could as well be dropped. It specifies the difference in the expected mass between snakes in region 2 and snakes in region 1. We need two parameters δ_k to specify the differences in the expected mass for snakes in habitats 2 and 3, respectively, relative to those in habitat 1.

We look at the design matrix

```
model.matrix(~region + hab)

> model.matrix(~region + hab)
  (Intercept)  region2  hab2  hab3
1           1        0     0     0
2           1        0     1     0
3           1        0     0     1
4           1        0     0     0
5           1        1     1     0
6           1        1     0     1
[ ... ]
```

Hence, the implied system of equations that the software solves for us is this:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta_2 \\ \delta_2 \\ \delta_3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Interestingly, there is no way to specify the main-effects model in a means parameterization.

Next, consider the model with interactive effects, which lets the effect of one factor level depend on the level of the other factor. The default effects parameterization is written like this in R:

```
lm(mass ~ region * hab)
> lm(mass ~ region * hab)

Call:
lm(formula = mass ~ region * hab)

Coefficients:
(Intercept)  region2   hab2   hab3   region2:hab2   region2:hab3
6.5          6.0      1.5     1.5    5.0            NA
```

We see that one parameter is not estimable. The reason for this is that in the 2-by-3 table of effects of region crossed with habitat, we lack snake observations for habitat 1 in region 2.

Algebraically, we assume that the mass of individual i in region j and habitat k can be broken down as follows:

$$\text{mass}_i = \alpha + \beta_{j(i)} * \text{region}_i + \delta_{k(i)} * \text{hab}_i + \gamma_{jk(i)} * \text{region}_i * \text{hab}_i + \varepsilon_i$$

with

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

In this equation, the meanings of parameters α, β_j, and δ_k remain as before, i.e., they specify the main effects of the levels for the habitat and region factors. The new coefficients, γ_{jk}, of which there are two, specify the *interaction effects* between these two factors.

Here is the design matrix for this model:

```
model.matrix(~region * hab)

> model.matrix(~region * hab)
  (Intercept)  region2  hab2  hab3  region2:hab2  region2:hab3
1           1        0     0     0             0             0
2           1        0     1     0             0             0
3           1        0     0     1             0             0
4           1        0     0     0             0             0
```

```
5          1        1      1      0          1          0
6          1        1      0      1          0          1
[...]
```

And here is, therefore, the system of equations that needs to be solved to get the parameter estimates for this model:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta_2 \\ \delta_2 \\ \delta_3 \\ \gamma_{22} \\ \gamma_{23} \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Finally, the means parameterization of the interaction model:

```
lm(mass ~ region * hab 1 region hab)
> lm(mass ~ region * hab 1 region hab)

Call:
lm(formula = mass ~ region * hab  1  region  hab)

Coefficients:
reg1:hab1  reg2:hab1  reg1:hab2  reg2:hab2  reg1:hab3  reg2:hab3
6.5        NA         8.0        9.0        5.0        11.0
```

(I slightly edited the output so it fits a single line.)

We see again that in the interactive model, we have no information to estimate the expected body mass of snakes in habitat of type 1 in region 2, since no snakes were examined for that combination of factor levels (called a *cell* in the cross-classification of the two factors). In the additive model, the information for that cell in the table comes from the other cells in the table, but in the interactive model, each cell mean is estimated independently.

This model can be written algebraically like this:

$$\text{mass}_i = \alpha_{jk(i)} * \text{region}_i * \text{hab}_i + \varepsilon_i$$

and

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

There are six elements in the vector α_{jk}, corresponding to the six ways in which the levels of the two factors `region` and `habitat` can be combined. Also note that $region_i * hab_i$ implies six columns in the design matrix. Now look at the design matrix (again, slightly edited for an improved presentation):

```
model.matrix(~ region * hab 1 region hab)
> model.matrix(~ region * hab 1 region hab)
  reg1:hab1  reg2:hab1  reg1:hab2  reg2:hab2  reg1:hab3  reg2:hab3
1  1          0          0          0          0          0
2  0          0          1          0          0          0
```

```
3  0        0        0        0        1        0
4  1        0        0        0        0        0
5  0        0        0        1        0        0
6  0        0        0        0        0        1
[ ... ]
```

And the system of equations:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{12} \\ \alpha_{22} \\ \alpha_{13} \\ \alpha_{23} \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

We see clearly the lack of information about effect α_{21}, which is the expected mass of a snake in region 2 and habitat 1, represented by the second column in the design matrix containing all zeroes.

6.3.6 Analysis of Covariance

When we are interested in the effects on `mass` of both a factor (= discrete explanatory variable, e.g., `pop`) and a continuous covariate like `svl`, we could specify an ANCOVA model (see also Chapter 11). There are two ways in which we might like to specify that model. First, we may think that the relationship between `mass` and `svl` is the same in all populations, or worded in another way, that the mass differences among populations do not depend on the length of the snake. In statistical terms, this would be represented by a *main-effects model*. Second if we admitted that the mass–length relationship might differ among populations or that the differences in mass among populations might depend on the length of a snake, we would fit an *interaction-effects model*. (Note that here "effects" has a slightly different meaning from that when used as effects parameterization vs. means parameterization.) To specify these two versions of an ANCOVA in R, we write this:

```
lm(mass ~ pop + svl)          # Additive model
lm(mass ~ pop * svl)          # Interactive model
lm(mass ~ pop + svl + pop:svl) # Same, R's way of specifying the interaction term
```

Here's the additive model algebraically in the effects parameterization:

$$\text{mass}_i = \alpha + \beta_{j(i)} * \text{pop}_i + \delta * \text{svl}_i + \varepsilon_i,$$

with

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

This model says that the mass of snake i in population j is made up of a constant α plus the effects β_j when in populations 2 and 3 plus a constant δ times `svl` plus the residual. In the effects parameterization, α is the

intercept for population 1 and vector β_j has two elements, one for population 2 and another one for population 3, being the differences of the intercepts in these populations and the intercept in population 1. Finally, δ is the common slope of the mass–length relationship in all three populations.

The means parameterization of the same model is this:

$$\text{mass}_i = \alpha_{j(i)} * \text{pop}_i + \delta * \text{svl}_i + \varepsilon_i,$$

with

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

All that changes is that now the vector α_j has three elements representing the intercepts in each population.

The effects parameterization of the interaction-effects model is written like this:

$$\text{mass}_i = \alpha + \beta_{j(i)} * \text{pop}_i + \delta * \text{svl}_i + \gamma_{j(i)} * \text{svl}_i * \text{pop}_i + \varepsilon_i$$

and

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

In this model, α is the intercept for population 1 and vector β_j has two elements, representing the difference in the intercept between population 1 and populations 2 and 3, respectively. Parameter δ is the slope of the mass–length relationship in the first population. Vector γ_j has two elements corresponding to the difference in the slope between population 1 and populations 2 and 3, respectively.

The means parameterization of the same model is probably easier to understand and is written like this:

$$\text{mass}_i = \alpha_{j(i)} * \text{pop}_i + \delta_{j(i)} * \text{svl}_i + \varepsilon_i$$

and

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

The only change relative to the main-effects model is that we added a subscript j to the effect δ of svl, meaning that we now estimate three slopes δ, one for each population, instead of a single one that is common to snakes from all three populations.

Let's now look at the design matrices of both the effects and the mean parameterizations of both the main- and interaction-effects models. Here is the design matrix of the main-effects ANCOVA model using the R default, the effects parameterization (I do hope that this terminology is not too confusing here …):

```
model.matrix(lm(mass ~ pop + svl))    # Additive model
```

```
> model.matrix(lm(mass ~ pop + svl))
  (Intercept)  pop2  pop3  svl
1           1     0     0   40
2           1     0     0   45
```

```
3           1     1     0    39
4           1     1     0    50
5           1     0     1    52
6           1     0     1    57
[ ... ]
```

The first parameter, the intercept, signifies the expected mass in population 1 at the point where the value of the covariate is equal to zero, i.e., for a snake of length zero. That is, therefore, the intercept of the regression model. The parameters associated with the design matrix columns named `pop2` and `pop3` quantify the difference in the intercept between these populations and population 1. The parameter associated with the last column in the design matrix measures the common slope of `mass` on `svl` for snakes in all three populations.

And here is the design matrix of the interaction-effects ANCOVA model using the R default, the effects parameterization:

```
model.matrix(lm(mass ~ pop * svl))      # Interactive model

> model.matrix(lm(mass ~ pop * svl))
  (Intercept)     pop2     pop3     svl     pop2:svl     pop3:svl
1           1        0        0      40            0            0
2           1        0        0      45            0            0
3           1        1        0      39           39            0
4           1        1        0      50           50            0
5           1        0        1      52            0           52
6           1        0        1      57            0           57
[ ... ]
```

The parameters associated with the first three columns in this design matrix signify the population 1 intercept and the effects of populations 2 and 3, respectively, i.e., the difference in intercepts. The parameter associated with the fourth column, `svl`, is the slope of the mass–length regression in the first population, and the parameters associated with the last two columns are the differences between the slopes in populations 2 and 3 relative to the slope in population 1.

Now let's see the main-effects model using the means parameterization:

```
model.matrix(lm(mass ~ pop + svl 1))      # Additive model

> model.matrix(lm(mass ~ pop + svl 1))
  pop1  pop2  pop3  svl
1    1     0     0   40
2    1     0     0   45
3    0     1     0   39
4    0     1     0   50
```

```
5    0      0      1    52
6    0      0      1    57
[ ... ]
```

The parameters associated with the first three columns in the design matrix now directly represent the intercepts for each population, while that associated with the fourth column denotes the common slope of the `mass svl` relationship.

And finally, let's formulate the interaction-effects model using the means parameterization.

```
model.matrix(lm(mass ~ (pop * svl  1  svl)))     # Interactive model
```

```
> model.matrix(lm(mass ~ pop * svl 1))
  pop1  pop2  pop3  pop1:svl   pop2:svl  pop3:svl
1    1     0     0        40          0         0
2    1     0     0        45          0         0
3    0     1     0         0         39         0
4    0     1     0         0         50         0
5    0     0     1         0          0        52
6    0     0     1         0          0        57
[ ... ]
```

This parameterization of the ANCOVA model with interaction between population and svl contains parameters that have the direct interpretation as intercepts and slopes of the three `mass svl` relationships (one in each population). We will see this below, where we will also see that in R we can directly fit in an `lm()` function an object that is a design matrix.

Here is the matrix–vector description of the main-effects ANCOVA model with effects parameterization applied to our snake data set:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 40 \\ 1 & 0 & 0 & 45 \\ 1 & 1 & 0 & 39 \\ 1 & 1 & 0 & 50 \\ 1 & 0 & 1 & 52 \\ 1 & 0 & 1 & 57 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \delta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

And here is the means parameterization of the main-effects ANCOVA:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 40 \\ 1 & 0 & 0 & 45 \\ 0 & 1 & 0 & 39 \\ 0 & 1 & 0 & 50 \\ 0 & 0 & 1 & 52 \\ 0 & 0 & 1 & 57 \end{pmatrix} * \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \delta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Here is the effects parameterization of the interactive model:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 40 & 0 & 0 \\ 1 & 0 & 0 & 45 & 0 & 0 \\ 1 & 1 & 0 & 39 & 39 & 0 \\ 1 & 1 & 0 & 50 & 50 & 0 \\ 1 & 0 & 1 & 52 & 0 & 52 \\ 1 & 0 & 1 & 57 & 0 & 57 \end{pmatrix} * \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \delta \\ \gamma_1 \\ \gamma_2 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

And finally the means parameterization of the same model:

$$\begin{pmatrix} 6 \\ 8 \\ 5 \\ 7 \\ 9 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 40 & 0 & 0 \\ 1 & 0 & 0 & 45 & 0 & 0 \\ 0 & 1 & 0 & 0 & 39 & 0 \\ 0 & 1 & 0 & 0 & 50 & 0 \\ 0 & 0 & 1 & 0 & 0 & 52 \\ 0 & 0 & 1 & 0 & 0 & 57 \end{pmatrix} * \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \delta_1 \\ \delta_2 \\ \delta_3 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

Now let's use R to fit the main-effects and the interaction-effects models using both the effects parameterization and the means parameterization. First, R's default effects parameterization of the main-effects model:

```
lm(mass ~ pop + svl)
> lm(mass ~ pop + svl)

Call:
lm(formula = mass ~ pop + svl)

Coefficients:
(Intercept)         pop2         pop3          svl
    3.43860      1.49123      0.05263      0.24561
```

So, the intercept is the expected mass of a snake in population 1 that has `svl` equal to zero. `Pop2` and `pop3` are the differences in the intercept between populations 2 and 3 compared with that in population 1, and the parameter named `svl` measures the slope of the `mass svl` relationship common to snakes in all populations.

It is instructive to plot the estimates of the relationships for each population under this model (Fig. 6.5).

```
fm <- lm(mass ~ pop + svl)          # Refit model
plot(svl, mass, col=c(rep("red", 2), rep("blue", 2), rep("green", 2)))
abline(fm$coef[1], fm$coef[4], col = "red")
abline(fm$coef[1]+ fm$coef[2], fm$coef[4], col = "blue")
abline(fm$coef[1]+ fm$coef[3], fm$coef[4], col = "green")
```

This model assumes that the `mass svl` relationship differs among populations only in the average level. What we see then is that population 1 (red) hardly differs from population 3 (green), but that snakes in

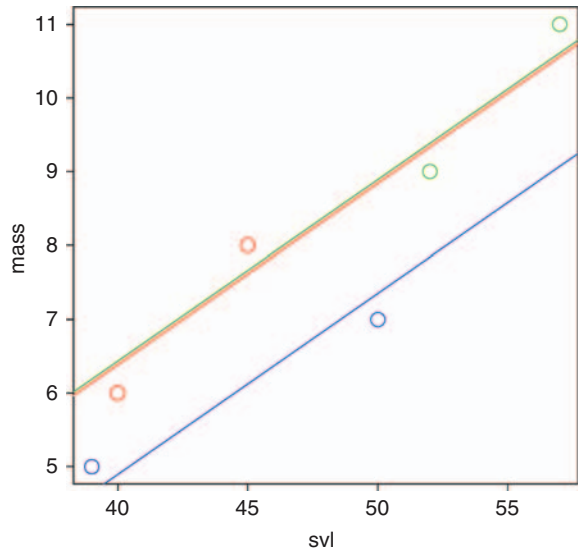

FIGURE 6.5 The main effects ANCOVA model.

population 2 (blue) weigh less at a given length than do snakes in population 1 or 3.

Next, the interaction-effects models using the R default effects parameterization:

```
lm(mass ~ pop * svl)
> lm(mass ~ pop * svl)

Call:
lm(formula = mass ~ pop * svl)

Coefficients:
(Intercept)      pop2      pop3       svl   pop2:svl   pop3:svl
  1.000e+01 7.909e+00  1.800e+00 4.000e 01  2.182e 01  6.232e 17
```

The first and the fourth parameters describe intercept and slope of the relationship between `mass` and `svl` in the first population, while the remainder refer to intercept and slope differences between the other two populations and those of the baseline population. Note that the last parameter estimate should in fact be zero but is not due to rounding error.

We plot the estimated relationships also under the interaction model (Fig. 6.6):

```
fm <  lm(mass ~ pop * svl)            # Refit model
plot(svl, mass, col  c(rep("red", 2), rep("blue", 2), rep("green", 2)))
abline(fm$coef[1], fm$coef[4], col = "red")
```

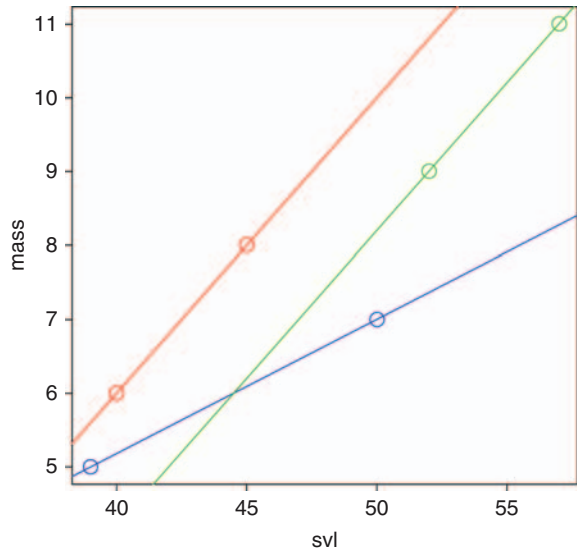

FIGURE 6.6 The interaction effects ANCOVA model.

```
abline(fm$coef[1]+ fm$coef[2], fm$coef[4] + fm$coef[5], col = "blue")
abline(fm$coef[1]+ fm$coef[3], fm$coef[4] + fm$coef[6], col = "green")
```

As an aside, this is not a useful statistical model, since it has an $R^2 = 1$, but the example does serve to illustrate the two kinds of assumptions about homogeneity or not of slopes that one may examine in an ANCOVA model.

Next, we try the means parameterizations of both the main-effects model and the interaction-effects model. First the main-effects model:

```
lm(mass ~ pop + svl 1)
> lm(mass ~ pop + svl 1)

Call:
lm(formula = mass ~ pop + svl  1)

Coefficients:
   pop1     pop2     pop3      svl
 3.4386   4.9298   3.3860  0.2456
```

This gives us the estimates of each individual slope plus the slope estimate common to all three populations.

What about the interaction-effects model?

```
lm(mass ~ pop * svl 1  svl)
> lm(mass ~ pop * svl 1  svl)
```

```
Call:
lm(formula = mass ~ pop * svl - 1 - svl)

Coefficients:
   pop1       pop2       pop3  pop1:svl  pop2:svl  pop3:svl
 10.0000    2.0909   11.8000    0.4000    0.1818    0.4000
```

These estimates have direct interpretations as the intercept and the slope of the three regressions of `mass` on `svl`.

6.4 SUMMARY

We have briefly reviewed the two key components of linear statistical models: statistical distributions and the linear predictor, which is represented by the product of the design matrix and the parameter vector. Understanding both is essential for applied statistics. But while sometimes one may get away in R or other useful stats packages with not exactly knowing what parameterization of a model is fit by the software and what the parameters effectively mean, this is not the case when using WinBUGS. In WinBUGS, we have to specify all columns of the design matrix and thus must know exactly what parameterization of a model we want to fit and how this is done. The linear models in this chapter were presented in a progression from simple to complex. Chapters 7–11 follow that structure and show how to fit these same models in WinBUGS and also in R for normal responses, i.e., for normal linear models, to which ANOVA, regression, and related methods all belong.

EXERCISE

1. *Fitting a design matrix*: The interaction-effects ANCOVA wasn't a useful statistical model for the toy snake data set, since six fitted parameters perfectly explain six observations and we can't estimate anymore the variability in the system. Use `lm()` to fit a custom-built design matrix, i.e., the design matrix of an ANCOVA with partial interaction effects, where the slopes of the mass length relationship are the same in population 1 and population 3. Build this design matrix in R, call it `X`, and fit the model by directly specifying `X` as the explanatory variable in function `lm()`.