CHAPTER

WHAT ARE HIERARCHICAL MODELS AND HOW DO WE ANALYZE THEM?

2

CHAPTER OUTLINE

Applied Hierarchical Modeling in Ecology. http://dx.doi.org/10.1016/B978-0-12-801378-6.00002-3

Copyright © 2016 Published by Elsevier Inc. All rights reserved.

2.1 INTRODUCTION

In this chapter we provide a conceptual definition of the term *hierarchical model* (HM) as we use it throughout the book. We use the term to describe a coupled set of ordinary (or "flat") models that are conditionally related to each other. By conditional we mean in the sense of conditional probability: the probability distribution of one random variable depends on the other random variable, usually expressed as $f(y|z)$ or $[y|z]$, indicating that probabilities of outcomes of the random variable y depend on the outcome of the random variable z. Typically the models contain one component for the observations, or data, and then one or more additional components to describe latent variables or outcomes of some ecological process. In "integrated models" (Chapter 23), we have more than one observation model, one for each type of data. While our usage of the term HM is somewhat more specific than it is used in the statistical literature, we find that it serves to provide a conceptual unification of many different types of models, such as occupancy models (MacKenzie et al., 2002), *N*-mixture models (Royle, 2004b), distance sampling (Royle et al., 2004, Sillett et al., 2012), and many others (Royle and Dorazio, 2008; Kéry and Schaub, 2012).

One of the overarching themes of this book is the analysis of HMs using both classical and Bayesian inference methods. We do this because we think that in order to be an effective modeler, you need to understand and be able to apply both, depending on the situation. There will be some problems that are most easily, or perhaps only, solvable using Bayesian analysis—by Markov chain Monte Carlo (MCMC) methods—and other problems that are best solved using classical likelihood methods. For instance, models that have spatially correlated random effects cannot easily be analyzed using classical likelihood methods, while many standard discrete mixture models such as the *N*-mixture are easily analyzed by likelihood methods, and therefore we can do things like Akaike's information criterion (AIC)-based model selection.

What do we mean by "classical inference"? By that we mean, strictly speaking, direct analysis of the likelihood; e.g., estimation of parameters by maximum likelihood. We usually also mean the conceptual view of *frequentist* inference, in which the properties of estimators and procedures are evaluated by thinking about averaging over possible realizations of data. Therefore, in a straightforward practical

sense, we mean likelihood estimation but we imagine that the frequentist evaluation of things is implied, especially as it relates to the consistency of maximum likelihood estimates (MLEs) or the interpretation of confidence intervals. Bayesian inference is becoming more and more familiar to practicing ecologists (McCarthy, 2007; Royle and Dorazio, 2008; Kéry, 2010; Kéry and Schaub, 2012; Hobbs and Hooten, 2015). The principal difference between Bayesian and classical inference has to do with how random variables are used to formulate models. In particular, classical inference regards parameters as fixed but unknown quantities, whereas Bayesians regard parameters as random variables endowed with a prior distribution, and as a result this enables the Bayesian to base inference on the probability distribution of the unknown quantity *given* the data; i.e., on the posterior distribution. Practical Bayesian analysis is now almost always carried out using MCMC methods that we introduce here, including some basic principles for constructing MCMC algorithms (in Chapter 5 we show how to do these things with BUGS software, and discuss some practicalities of using MCMC methods, such as convergence diagnostics and so forth). Finally, we discuss the important topics of model selection and model checking (or goodness-of-fit assessment) from both classical and Bayesian perspectives.

2.2 RANDOM VARIABLES, PROBABILITY DENSITY FUNCTIONS, STATISTICAL MODELS, PROBABILITY, AND STATISTICAL INFERENCE

Although not frequently taught to ecologists, random variables and probability density/mass functions are essential for statistical modeling. So what are they? A variable is a characteristic or feature that exhibits variability among units or elements that possess that characteristic or feature. For example, the height of an adult male, the number of peregrines counted during the spring peregrine survey, or the color of an M&M that I'm about to pull from a bag and eat. In other words, variables are quantities that can take on different values. *Random* variables are variables for which possible values are governed by probability distributions. The precise manner in which possible values are characterized is called the *probability density function* (pdf) if the variable is continuous (height of adult males) and it is called the *probability mass function* (pmf) if the variable is discrete (peregrine counts or colors of M&Ms). The pmf or pdf is a mathematical rule that governs possible outcomes of a random variable. Usually, the pmf (or pdf) depends on one or more quantities, called *parameters*, that affect its characteristic form. For example, one of our favorites is the binomial pmf for counts y with a natural upper limit N (representing, for instance, the number of samples or replicates or a population size):

$$f(y) = \frac{N!}{y!(N-y)!} p^y (1-p)^{(N-y)} \tag{2.1}$$

where y can take on integer values from $y = 0$ to $y = N$, and the single parameter p is known as the "success probability" which will typically be detection or encounter probability in many ecological sampling applications. This binomial pmf tells us the probabilities of each possible value of y, and so sometimes instead of $f(y)$ we will denote a pmf by writing $\Pr(y)$, $y \sim Binomial(p, N)$, or $y \sim Bin(p, N)$, where the $\sim$ is usually read "is distributed as." As an example, if the random variable is $y =$ "the number of times I observed at least one peregrine falcon" (Figure 2.1) during $N = 5$ visits to a nesting cliff, and I know that the value of the parameter $p = 0.2$, then the possible values of the random variable are $y = \{0,1,2,3,4,5\}$, and they occur with the probabilities given by Eq. (2.1), which in the R language can be computed as follows:

```
dbinom(0:5, size = 5, prob = 0.2)
[1] 0.32768 0.40960 0.20480 0.05120 0.00640 0.00032
```

FIGURE 2.1

The magnificent peregrine falcon (*Falco peregrinus*): adult female with fledged young in the Jura Mountains, Switzerland, 2008. *(Photo: Alain Georgy.)*

What this translates to, in words, is that the probability that I detect peregrines zero times is 0.32768, the probability that I detect peregrines one time is 0.40960, and so forth. We can obtain the probability of getting any number in a range of numbers, such as 2, 3, or 4, by summing the associated probabilities of each event in the set; e.g., $\Pr(y = \{2,3,4\}) = 0.20480 + 0.05120 + 0.00640 = 0.2624$.

The pdf is a similar rule but for continuous random variables. Continuous random variables have the peculiar feature that the probability of observing any *particular* value of the variable is 0 (and yet, in practice, we may observe some value multiple times but still use a continuous variable as a model). Thus, a sensible interpretation of a pdf is that it gives the probabilities of values that lie in prescribed intervals. One of the standard pdf's that we will use is the normal pdf that has this form:

$$f(y) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right)$$

The normal probability distribution has two parameters, the mean μ and the standard deviation σ. If the height of adult males in a well-defined population is known to have mean $\mu = 190$ cm and standard deviation $\sigma = 10$ cm, then we can compute probabilities for specific ranges of y by integrating $f(y)$ over the desired range. For example, the probability that y is *less than* some value L is computed by integrating $f(y)$ from $-\infty$ to L, the probability that y lies between two values L and U is the integral

$\int_L^U f(y)dy$, and so forth. In R we can use the `pnorm` function to do these kinds of operations for us. To illustrate, the probability that $y < 200$ in this hypothetical population is:

```
pnorm(200, mean = 190, sd = 10)
[1] 0.8413447
```

So we expect that slightly more than 84% of the population is shorter than 200 cm. Or, the expected proportion of males with height between 180 and 200 (i.e., between -1 and $+1$ standard deviation (SD) of the mean) is:

```
pnorm(200, mean = 190, sd = 10) - pnorm(180, mean = 190, sd = 10)
[1] 0.6826895
```

You can compute this result directly by integrating the normal pdf using the R function `integrate` as follows:

```
f <- function(x, mu, sigma){
  (1 / sqrt(2*pi*sigma^2)) * exp( -((x-mu)^2)/(2*sigma^2))
}
integrate(f, lower = 180, upper = 200, mu = 190, sigma = 10)
0.6826895 with absolute error < 7.6e-15
```

The `integrate` function is useful in a lot of routine analyses of HMs that we do, although often its usage is hidden "under the hood" in R packages such as `unmarked` (Fiske and Chandler, 2011).

We will often use the shorthand "bracket notation" to represent probability distributions. For example, $[y]$ is the marginal distribution of the random variable y, $[y,x]$ is the joint distribution of y and x, and $[y|x]$ is the conditional distribution of y given x. See also Section 2.2.2.

Some statistical distributions have a parameter that directly represents the mean expected response (e.g., the normal has parameter μ, and the Poisson has parameter λ), while some have a parameter that represents the variability, "spread," or dispersion of the response (e.g., σ in the normal). This allows us to model the mean tendency of a random variable or its spread by using covariates or other model structures such as random effects in a statistical model. However, for most distributions the mean and the dispersion are *not* represented by single parameters, but by functions of one or more parameters, such as Np for the binomial mean or $a/(a + b)$ for the mean of a beta distribution. The standard pdf can often be rewritten (or reparameterized) such that the mean (or the variance) of the response becomes a parameter that can then be modeled as a function of covariates (and adding of covariates is much of what we do when we do "statistical modeling"). This reparameterizing is called "moment matching" (Hobbs and Hooten, 2015), and can be very useful in modeling (see 5.12 and 7.6 for examples). A large number of statistical distributions, along with their pdf's (or pmf's) and the expressions for their means and variances, can be found on the Internet or in any number of statistics texts. These are the main building blocks of your statistical models (Bolker, 2008).

2.2.1 STATISTICAL MODELS

In most statistical models, a measured response is treated as a random variable endowed with a pdf that is assumed known. Hence, a statistical model for observed data is represented by a pdf. The pdf accommodates both the randomness (the unpredictable part) in the response and also the tendency for some values to occur more frequently than others—i.e., the part in the response that is predictable at

least in an average sense. Almost always, we have some other variables that we assume cause some of the variability in the response, or at least are associated with the response in some stochastically predictable way. In the statistical model, the basic parameters of the pdf are then replaced by some linear or nonlinear function of these *covariates*.

Explicit specification of pdfs and pmfs may not be the typical description of a statistical model that you have seen previously in your training as an ecologist. In fact, it is quite likely that in your statistics classes at university you have *never* even encountered an explicit statistical model (i.e., one written as a pdf), but rather you had to memorize a certain range of study situations or data collection protocols that led to certain procedures that produced some numbers such as *p-values*. However, random variables and pdf's are the statisticians' way to think about and describe statistical models, and it is this way that ecologists also should think about statistical models. In part this is because viewing a model as a description of a single thing that is variable—i.e., a random variable described by a pdf—leads naturally to the adoption of HMs, where more than a single thing can vary simultaneously. In HMs, you will combine two or more random variables in sequence in a manner consistent with your scientific understanding of the modeled process, leading to what some have called a "scientific model" (Berliner, 1996). Hierarchical modeling represents a manner of "learning from data" that has tremendous power and considerable beauty, as we will see throughout this book.

2.2.2 JOINT, MARGINAL, AND CONDITIONAL DISTRIBUTIONS

In developing HMs, we are always interested in two or more random variables and how they are related to one another. Inferences about relationships among these random variables can be made using joint, marginal, or conditional probability distributions depending on the question being asked (note that much of the following material is modified from Chapter 2 in Royle et al., 2014). For our immediate purpose here of developing ideas related to these probability distributions, we will deviate slightly from our previously established notation by distinguishing between random variables and *realizations* of random variables. That is, let Y be a random variable, and let y indicate a particular realization of the random variable. Therefore we make statements such as $\Pr(Y = y)$ and so forth. This distinction is customarily done in classical statistical training, but in practice it is efficient to ignore it when the specific context is clear (which we do elsewhere in this book).

Conditional probability distributions provide an explicit description of the relationship between one random variable, say Y, and some other random variable, say X (or sometimes multiple other variables). Joint distributions describe the probabilities of simultaneously observing all unique combinations of both Y and X. This is easiest to conceptualize in the case of discrete random variables, where the joint distribution is the probability that X takes on the value x *and* that Y takes on the value y, which is written $[Y = y, X = x]$. The marginal distribution of a variable Y is its distribution "averaged over" all possible values of X. When X is a discrete random variable, we formally carry out this averaging, whereas if X is continuous, this operation is equivalent to integration over the range of X (see below).

To clarify these concepts of conditional, joint, and marginal distributions in more concrete terms, let us revisit our example where Y is the number of peregrine sightings after $N = 5$ survey visits to a nesting cliff, which we assumed to be a binomial random variable. Now, let us suppose that Y depends on the random variable X, which is the number of fledged young at a nesting cliff at the time of survey (assuming that all surveys take place after the young have fledged). Specifically, let us say that the probability of observing a peregrine, p, is related to a realized value of the random variable X (i.e., the value $X = x$), according to $\text{logit}(p) = -1.2 + 2x$. Furthermore, let us make the intuitive assumption

that the number of fledged young around the cliff is a Poisson random variable with mean 0.4—i.e., $X \sim Poisson(0.4)$. Our model is now fully specified, and so we can answer the question: "What is the probability of observing peregrines y times *and* of there being x fledged young at the cliff?" This question may not be of interest by itself, but once we know how to answer this question we can answer much more useful questions as well, as we demonstrate below.

This *joint distribution* of Y and X is given by the product of the binomial pmf (with p determined by x) and the Poisson pmf with $\lambda = 0.4$. The following R commands create the joint distribution:

```
Y <- 0:5  # Possible values of Y (# surveys with peregrine sightings)
X <- 0:5  # Possible values of X (# fledged young)
p <- plogis(-1.2 + 2*X) # p as function of X
round(p, 2)

[1] 0.23 0.69 0.94 0.99 1.00 1.00

# Joint distribution [Y, X]
lambda <- 0.4
joint <- matrix(NA, length(Y), length(X))
rownames(joint) <- paste("y=", Y, sep="")
colnames(joint) <- paste("x=", X, sep="")
for(i in 1:length(Y)) {
  joint[,i] <- dbinom(Y, 5, p[i]) * dpois(X[i], lambda)
}

round(joint, 3)

      x=0   x=1   x=2   x=3   x=4 x=5
y=0 0.180 0.001 0.000 0.000 0.000   0
y=1 0.271 0.009 0.000 0.000 0.000   0
y=2 0.163 0.038 0.000 0.000 0.000   0
y=3 0.049 0.085 0.001 0.000 0.000   0
y=4 0.007 0.094 0.012 0.000 0.000   0
y=5 0.000 0.042 0.040 0.007 0.001   0
```

This matrix tells us the probability of all possible combinations of Y and X, and we see that the most likely outcome is $(y = 1, x = 0)$, which is to say that we are likely to detect peregrines only one time in our five visits to the nesting cliff, and the cliff is most likely to have zero fledged young (i.e., reproduction was not successful that year). Perhaps most peregrine watchers don't care about joint distributions, but a question that they might care about is whether they will have a successful peregrine watching trip, which we might phrase as the question "What is the probability of observing peregrines during all five days of our peregrine watching trip?" We know that this depends on the number of fledged young at a nesting cliff, but we don't know how many there are in advance, so this is a different question than "what are the most likely values of Y and X." This leads us to the marginal distribution, which is defined formally by

$$[Y] = \sum_X [Y, X] \qquad [X] = \sum_Y [Y, X]$$

for discrete random variables, and

$$[Y] = \int_{-\infty}^{\infty} [Y, X]\mathrm{dX} \qquad [X] = \int_{-\infty}^{\infty} [Y, X]\mathrm{dY}$$

for continuous random variables. The key idea here is that to get the marginal distribution of Y, we have to contemplate all possible values of X and average over them. Computing marginal distributions is a key step in maximizing likelihoods involving random effects, as we will discuss in Section 2.4.4. Here is some R code to compute the marginal distribution of X, the number of fledged young,

```
margX <- colSums(joint)
round(margX, 4)
   x=0    x=1    x=2    x=3    x=4    x=5
0.6703 0.2681 0.0536 0.0072 0.0007 0.0001
```

and for Y (i.e., the probability of seeing peregrines in $Y = y$ visits to the nesting cliff):

```
margY <- rowSums(joint)
round(margY, 4)
   y=0    y=1    y=2    y=3    y=4    y=5
0.1805 0.2792 0.2012 0.1352 0.1140 0.0899
```

The marginal pmf tells us that the most likely number of visits in which we will be successful at seeing peregrines is $y = 1$. We stand only a slim chance of seeing peregrines on all five visits to the cliff ($\Pr(Y = 5) = 0.0899$), and the probability that we see any peregrines at all is

$$\Pr(Y > 0) = 1 - \Pr(Y = 0) = \Pr(Y = 1) + \Pr(Y = 2) + \Pr(Y = 3) + \Pr(Y = 4) + \Pr(Y = 5) = 0.82.$$

We now turn to ideas of conditional probability, which we are interested in because in every application of hierarchical modeling, our model will involve an explicit specification of one or more *conditional* probability distributions. In a sense, conditional probability distributions are the elemental building blocks of HMs. In almost all applications we discuss in this book, we will find that certain natural conditional probability distributions lead to natural HMs for specific ecological problems. The conditional probability distribution is the distribution of one random variable, given (or conditional on) the realized value of the other. It is most easy to see how this is defined for the case of two discrete random variables. In that case, the conditional distribution may be written as $[Y = y|X = x]$—i.e., the probability of Y taking on the value y given the realized value of X being x. For simplicity, we will write this as $[Y|X]$, or most often for the rest of the book besides this chapter, we will ignore the upper- and lowercase convention and just write $[y|x]$. Conditional distributions are defined as follows:

$$[Y|X] = \frac{[Y, X]}{[X]} \qquad [X|Y] = \frac{[Y, X]}{[Y]}$$

That is, the conditional distribution of Y given X is the joint distribution divided by the marginal distribution of X.

```
YgivenX <- joint / matrix(margX, nrow(joint), ncol(joint), byrow=TRUE)
round(YgivenX, 2)
```

```
     x=0  x=1  x=2  x=3  x=4 x=5
y=0 0.27 0.00 0.00 0.00 0.00   0
y=1 0.40 0.03 0.00 0.00 0.00   0
y=2 0.24 0.14 0.00 0.00 0.00   0
y=3 0.07 0.32 0.03 0.00 0.00   0
y=4 0.01 0.35 0.23 0.04 0.01   0
y=5 0.00 0.16 0.74 0.96 0.99   1
```

Note that we have six probability distributions for Y, one for each possible value of X, and each pmf (column) sums to 1 as it should. In terms of its importance to bird watchers, we note that if there are >3 fledged young at this cliff, then you will expect to see peregrines nearly every day. The concepts of joint, marginal, and conditional distributions are explained in more detail in many other texts, Casella and Berger (2002) being one of our favorites.

The last point we wish to make in this section is that this made-up peregrine example *is* an HM, and we can put the pieces together using the following notation:

$$
\begin{aligned}
&X \sim Poisson(0.4) && \text{\# Submodel for no. young}\\
&\text{logit}(p) = -1.2 + 2X && \text{\# Relationship p and X}\\
&Y|X \sim Binomial(5, p) && \text{\# Submodel for observation}
\end{aligned}
$$

From here on out, when you see such notation, you should immediately grasp the fact that X is a random variable independent of Y, but Y depends on X through the parameter p. Now you have the tools to make probability statements about the random variables in this system. The one caveat faced in reality is that we typically do not know the values of the parameters, and instead we have to estimate them, either by classical methods such as the MLE, or by Bayesian methods, both of which we address in this chapter.

2.2.3 STATISTICAL INFERENCE

An important use of probability, as introduced in the previous section, is for providing a description of natural systems using random variables and probability distributions. More specifically, in this book we are concerned with using random variables to describe outcomes of ecological processes (observed and unobserved) and probability distributions as models for the variability in these possible outcomes. Probability therefore forms our conceptual basis for *modeling* in ecology.

On the other hand, the field of *statistics* is concerned with the basic problem of learning about the parameters of probability distributions from observed outcomes (i.e., "data") of some variable. The two dominant paradigms of statistical inference, Bayesian and classical, share a common conceptual linkage as being distinct flavors of parametric inference. This means that both require that we make explicit probability model assumptions about the random variables that describe our system. Either paradigm then proceeds to carry out some inference task (estimation, prediction, model selection, testing) that *assumes those parametric assumptions are truth*.

In the following sections, we discuss basic ideas of both likelihood and Bayesian inference and then demonstrate their application to HMs.

2.3 HIERARCHICAL MODELS (HMs)

As a very general definition we can say that a hierarchical model, which we typically abbreviate as "HM" in this book, is a sequence of related models ordered by their conditional probability structure. What this means practically is that HMs have one or more "intermediate" models/levels/stages involving a latent variable (or random effect). By this definition, classical random effects models are clearly HMs. As an example, imagine a survey on a set of $i = 1, 2, \ldots, M$ spatial units, wherein we take $j = 1, 2, \ldots, J$ replicate observations y_{ij} of some ecological measurement. Naturally we might suppose that replicate measurements on the same unit are more similar than those among units, and this is the standard motivation for allowing for a random group effect α. We might therefore specify the model for the observations according to

$$y_{ij}|\alpha_i \sim Normal(\mu + \alpha_i, \sigma^2)$$

where the random effect α_i also has a distribution such as:

$$\alpha_i \sim Normal(0, \sigma^2_\alpha)$$

Note that we emphasized the conditional dependence of y on α in the expression of the model for y. In shorthand, we use the bracket notation $[y|\alpha]$ to represent the observation model, where the vertical bar | indicates explicit conditioning of one variable (y in this case) on another (α in this case), and $[\alpha]$ represents the model for the random effect. The HM is therefore the pair of submodels $[y|\alpha]$ and $[\alpha]$. We note that there is not necessarily a logical "top-down hierarchy" involved in a hierarchical model, such as when we model individuals nested within populations etc. Hence, in a sense, an equally or perhaps even better fitting term might be "sequential model" (SM) instead of hierarchical model. But of course, the latter term is so much common-place that we wouldn't want to change it.

In this book, we usually use the term HM in a more specific context, in which the observation model is specified conditional on a latent variable that represents an actual (i.e., real) biological process or the outcome thereof. For example, we might condition our observed data y on the true (but unobserved) population size N of a sample unit. We view population size N as the outcome of a real ecological process, or more realistically the aggregate outcome of many ecological processes such as survival and recruitment. In that sense, N is very distinct conceptually from the latent group effect α in the preceding example, in the sense that in some cases N might be observable given sufficient effort, resources, and proper design, whereas α is never observable because it doesn't represent a real state of nature, but rather is a purely hypothetical construct. We call the model with α an "implicit HM" to distinguish it from the explicit HM in which N results from explicit ecological processes and has an explicit ecological meaning such as "abundance." Thus, while both the classical random effects model and that which is conditional on an explicit ecological process are "HMs," we find the second case to be more interesting and directly relevant to ecology, if we have a choice between the two. Nevertheless, our "explicit HMs" will sometimes also contain random group effects to accommodate groups and other factors inducing correlations.

We use a bit of jargon in our discussions of HMs, including words like *process* or *state model*, *state variable*, *observation model*, and *observations* (same as *data*). As used here, the *state variable* is the random variable described by the *state model* (N in the preceding paragraph), and the *observations* (y) are described by the *observation model*. In the next section, we will elaborate on and clarify some of these aspects of HMs by discussing two typical examples that are extremely relevant in ecology, since

they deal with distribution and abundance, which are defining concepts in ecology. Moreover, they lay the foundation for most models in this book.

2.3.1 TWO CANONICAL HMs IN ECOLOGY

We encounter a large number of distinct HMs in ecology, but two of the most prominent and widely used HMs are those for modeling species occurrence or distribution, often called "occupancy models" (MacKenzie et al., 2002; Tyre et al., 2003), and the *N*-mixture model for modeling species abundance (Royle, 2004b). We think of these as the foundational examples of "explicit" HMs because they are simple hierarchical extensions of ordinary (nonhierarchical, or "flat") models, logistic regression, and Poisson generalized linear models (GLMs), respectively, which are used throughout statistics to model ordinary binary and count data (see Chapter 3 for an applied introduction to GLMs).

2.3.1.1 The Occupancy Model for Species Distributions

We provide a brief introduction to the occupancy model (MacKenzie et al., 2002; Tyre et al., 2003), which is a natural HM for modeling species distributions (see also Chapter 10 and Kéry, 2011b). Imagine that a sample of M discrete sites is surveyed and the presence or absence of a species is noted. Let y_i denote the binary observations of presence/absence at site $i = 1, 2, \ldots, M$, where possible values are $y_i = 0$ ("absence") and $y_i = 1$ ("presence"). A key motivation for the development of occupancy models is that sometimes a species goes undetected even though it is present (in other words, the detection probability is <1). Thus, some of the absence *observations* should in practice correspond to instances of presence of the species. To specify the model for this situation, we introduce a new variable, denoted by z_i, which is the *true* presence/absence state of site i, where $z_i = 0$ is true absence, and $z_i = 1$ is true presence. This state variable is also binary.

The observation model links the observations to the state variable, and the standard observation model is a Bernoulli model with success probability $z_i p$:

$$y_i | z_i \sim Bernoulli(z_i p)$$

where p is the probability of detecting the species *given that it is present*. That is, if $z_i = 1$, then y_i is a Bernoulli trial with parameter p. Otherwise, y_i is a Bernoulli trial with probability 0, which is the same as saying that $y_i = 0$ with probability 1; i.e., if the species is truly absent, we must necessarily observe absence (we assume for now that the observers are well trained so that they do not misidentify the focal species, but see Chapter 19). The state model—i.e., the model for the ecological process of "true species presence or absence"—is similarly a simple Bernoulli model for a binary response:

$$z_i \sim Bernoulli(\psi_i)$$

where $\psi_i = \Pr(z_i = 1)$ is the probability of occurrence, or occupancy probability. Naturally, as this is itself an equivalent model to logistic regression (see Chapter 3), we model covariates that affect occurrence probability on the logit-linear scale (though see also 3.3.6). For example, if x_i is a measured habitat or landscape covariate for site i, then

$$\text{logit}(\psi_i) = \beta_0 + \beta_1 x_i.$$

As a practical matter, the parameter p and those of the model for z are not separately identifiable from a single observation of presence/absence at a site, unless we have some additional information. One way to obtain this information is through replicate surveys at each site, so that instead of a simple Bernoulli observation y_i, we have a sequence of repeated observations $y_{i1},\ldots,y_{iJ}$, where there are J sample

occasions. The standard assumption is that the sampling occurs over a sufficiently short period that a kind of "closure" with respect to occupancy status can be assured; i.e., occupancy status does not change over the J replicate surveys. In this case we can compute the total count, and for that we will abuse our notation and redefine y_i, so that now $y_i = \sum_{j=1}^{J} y_{ij}$, and the observation model is then a binomial model:

$$y_i|z_i \sim Binomial(J, z_i p)$$

This occupancy model has a transparent structure as a compound GLM—the model for the ecological state is a Bernoulli GLM and the model for the observations is a Bernoulli (or binomial) GLM *conditional* on the (in this case) partially observed state variable. This is arguably the simplest explicit HM imaginable because both the observations and the state variable are binary.

The occupancy model has become rather popular recently and is implemented in software MARK (White and Burnham, 1999), PRESENCE (Hines, 2006), and with the function `occu` in the R package `unmarked` (Fiske and Chandler, 2011). We discuss this model class in great detail in Chapters 10 and later.

2.3.1.2 The N-Mixture Model for Abundance

The N-mixture model (Royle, 2004b) is arguably the canonical HM for animal abundance. As for the occupancy model, the sampling design has an equivalent repeated measures structure where $i = 1, 2, \ldots, M$ sites are sampled for J occasions (e.g., point counts of birds on J mornings within the same season) and individuals of a species are counted. The observations are the counts y_{ij}, and the observation model is assumed to be a binomial distribution conditional on the true population size at site i:

$$y_{ij}|N_i \sim Binomial(N_i, p)$$

where p in this case is *individual*-level detection probability (as opposed to site-level detection in the case of the occupancy model). In general, we can model factors that influence detection using GLMs with the logit link function (see Chapter 3) but for now we neglect that generality. As with the occupancy model we just described, it is necessary to have repeated counts (= "measurements of abundance N") for at least some of the sites to ensure identifiability of model parameters.

The state variable here is "local population size," N_i, and the Poisson GLM is the natural framework for modeling variation in this state variable:

$$N_i \sim Poisson(\lambda_i).$$

In almost all studies there is interest in modeling the effect of measurable covariates, so we consider models for the log-transformed λ_i (i.e., the canonical link for Poisson GLMs; see Chapter 3):

$$\log(\lambda_i) = \beta_0 + \beta_1 x_i$$

where x_i is some site-level covariate. Of course, we may consider alternative models for local abundance such as the zero-inflated Poisson, Poisson lognormal, or negative binomial, all of which account for excess variation (or overdispersion) in local abundance relative to the Poisson model (Chapter 6 and Kéry et al., 2005b).

As with the occupancy model, the N-mixture model is clearly a type of compound GLM—both observation and process models are plain old GLMs, but they are linked together through the conditional dependence structure of the HM. The N-mixture models are implemented in program PRESENCE (Hines, 2006), program MARK (White and Burnham, 1999), and in `unmarked` with the function `pcount`, and they are the focus of a series of chapters starting with Chapter 6.

2.3.2 THE PROCESS OF HIERARCHICAL MODELING

HMs are the outcome of a certain process of building a model and a certain manner of thinking about the processes underlying observed data (Berliner, 1996). In ecological systems, we find that hierarchical modeling is a very appealing way to think about how to build models to idealized systems; i.e., simpler problems that we wish we had, and then linking these idealized models to observations. One way to think about constructing HMs is to ask this question: What unknown variable, if I had it, would simplify my problem? In all of the work we do, our attempt to answer this unifies the different HMs: (1) occupancy models—true occupancy state renders the model for observations as a simple logistic regression (2) *N*-mixture models—true population size renders the model for observations to be a simple logistic regression, and (3) spatial capture–recapture models (not covered here; see Royle et al., 2014)—the activity center renders the model for trap-level observations to be a simple logistic regression. Actually, the traditional binomial measurement error model underlying almost all capture–recapture and related models renders most of these models, at the level of the observed data, a type of logistic regression with a kind of complex random effects structure.

2.3.3 INFERENCE FOR HMs

HMs may be analyzed using both classical or Bayesian methods and either approach may be advantageous in some situations. Because HMs are specified conditional on one or more latent variables, Bayesian analysis proceeds directly with the use of Markov chain Monte Carlo (MCMC) algorithms that preserve the latent variables in the model. On the other hand, the classical method of likelihood inference for models with latent variables is based on marginal (or integrated) likelihood in which the latent variable is removed from the conditional likelihood (i.e., that containing the latent variable) by integration or, in the case of a discrete latent variable, by summation. In the following sections we address basic ideas of both classical and Bayesian inference.

2.4 CLASSICAL INFERENCE BASED ON LIKELIHOOD

Statistical inference for any system is based on the conceptual view that data we observe (or may potentially observe) are outcomes of random variables having a distribution with parameters that we would like to learn about from data. Most of the time we will denote the random variable as y, let us say the count of birds at some point count location. Absent some specific context, we will denote the basic probability model for y as follows:

$$y \sim f(y|\theta)$$

that we will read "the random variable y has probability distribution f," although normally the $\sim$ symbol is read "is distributed as" and we also usually do not say "the random variable." In this representation there is a parameter θ, or possibly a vector of parameters, that indexes the probability distribution f.

Once we go to a site and perform an actual point count of birds we obtain some realized value of this random variable, say $y = 5$ birds counted. The value $y = 5$ is our observation, data set, data, or a datum. To be completely clear, we must say this: When we use the $y \sim f(y|\theta)$ notation, we do not mean that the number 5 has the distribution f (which would be nonsense) but rather, and clearly, we mean that

we are using a random variable having distribution f as a model for potential outcomes of the count, including the outcome(s) we have on hand ($y = 5$ in this case). In classical statistics, as we have noted before, this distinction is usually made explicitly with uppercase and lowercase letters that we are not generally adhering to, possibly at the expense of some clarity in certain instances.

We usually consider the possibility that we might obtain outcomes of multiple random variables having this distribution f, say a sample of size n, and we denote these multiple random variables by $y_1, \ldots, y_n$ or just the vector $\mathbf{y}$ (bold type here indicates a vector). In this case we assert, as our objective, the need to estimate the parameter θ from this sample of size n.

It is not possible to proceed without additional assumptions. In particular, while f is the distribution of a single random variable y, it does not provide a characterization of the distribution of multiple random variables. For that, we need to build their *joint distribution*. If we assume that each random variable is independent of each other, then the joint distribution of several random variables is the product of the distribution of each one. Therefore, the joint distribution of our sample of n random variables is:

$$f(y_1, y_2, \ldots, y_n) = \prod_{i=1}^{n} f(y_i|\theta)$$

Identifying the joint distribution of the sample of n random variables (those which may be observed) is the crucial step toward achieving formal inference procedures about θ from a data set. We cannot do anything statistical in terms of estimation or any other inference without being able to identify what this joint distribution is. This task is made simpler by assuming independence of the observable random variables, which implies directly the joint distribution from the simpler more elemental distribution of a single observation $f(y|\theta)$. But there will not generally be a direct correspondence between saying "the distribution of y is $f()$" and the joint distribution we need in order to develop inferences from a sample of n observations but we will often assume that random variables are independent so that the joint distribution is implied directly. Or more generally, we make explicit conditional independence assumptions, which is to say that the random variables are independent of one another conditional on some other variable. For example, in the N-mixture model in which N is the population size of individuals exposed to sampling and y_1 and y_2 are consecutive counts (e.g., on different days), we assume that y_1 and y_2 are independent of each other *conditional on N*. Then, we can make progress toward analyzing the N-mixture model by working with the joint distribution of y_1 and y_2, conditional on N, which is $[y_1,y_2|N] = [y_1|N][y_2|N]$; i.e., it is the product of two binomials. This still depends on the unknown quantity N, and additional consideration needs to be paid to this (see Section 2.4.4).

Having identified the joint distribution of our observable random variables we can proceed with obtaining an estimate of θ using the *method of maximum likelihood*, or the maximum likelihood estimator (MLE) (Edwards, 1992; Chapter 7 in Casella and Berger, 2002). As its name suggests, this method is based on the *likelihood*. Simply put, the likelihood is the joint distribution of the observable random variables, but regarded as a function of the parameter θ. We will denote this by $L()$ such as:

$$L(\theta; \mathbf{y}) \equiv f(y_1, y_2, \ldots, y_n|\theta) = \prod_{i=1}^{n} f(y_i|\theta).$$

Here we put a ";" between the arguments of L to distinguish that this is a function indexed by $\mathbf{y}$ and *not* conditional on $\mathbf{y}$ in the sense of conditional probability.

The value of θ that produces the highest likelihood for the observed data set is called the MLE and denoted by the parameter with a "hat" over it, $\widehat{\theta}$. For practical (computational) reasons we often work in terms of the natural logarithm of the likelihood. We can do this because maximizing the likelihood or log-likelihood are equivalent operations, as is *minimizing* the *negative* of the log-likelihood. In the R software, the functions `nlm` or `optim` by default perform function minimization, and so you will see things multiplied by -1 in the definition of the likelihood function (see below).

The term "estimator" is an important one in statistics. An estimator is a rule for converting data to a guess for θ. This is different from an *estimate* that is a specific numerical value produced by the estimator. Estimators have statistical properties, so we say things about bias and variance when we talk about the rules (estimators) we use to produce estimates (specific guesses). We can think about an estimator in a more tangible way as an R function that takes data and returns some summary of the data. In that context then, we can evaluate the properties of the R function by simulating many data sets and executing the R function over and over again for each data set. The numerical result of the R function in some particular case is the estimate, while the function itself is the estimator. Note that strictly it is not correct to speak about an "unbiased estimate," because only the estimator can have the property of zero bias, not a single output from the estimator. However, we may sometimes see "unbiased estimate" as a short and somewhat colloquial way of saying "the estimate produced by an unbiased estimator."

2.4.1 THE FREQUENTIST INTERPRETATION

Classical inference, including inference based on likelihood, is justified by frequentist evaluation of statistical estimators. That is, the performance of an estimator (or other procedure) is evaluated by "averaging over" hypothetical realizations of the data y. For example, if $\widehat{\theta}$ is an estimator of θ then the frequentist is interested in properties of $\widehat{\theta}$ over replicate realizations of the data. The expected value, $\mathrm{E}_y(\widehat{\theta}|y)$, is used to characterize bias: If the expected value is equal to θ, then $\widehat{\theta}$ is unbiased. The "sampling variance" of an estimator,$\mathrm{Var}_y(\widehat{\theta}|y)$, is what we use to characterize uncertainty, or precision, of MLEs. The variance is therefore taken with respect to hypothetical realizations of the data. The square root of the sampling variance is the standard error of an estimator (which may sometimes be a variance parameter, in which case we have a standard error of a variance).

In adopting a frequentist view of statistical inference (which is usually implicit by our adoption of MLE and asymptotic variances) we are not using probability directly to make inference about θ, and so we cannot make direct probability statements about θ. Instead, the probability statement is about the procedure itself. We see this in the standard explanation of a confidence interval which is "the probability that the interval contains the true value of the parameter is 95%"; i.e., it is a statement about the interval itself and not the parameter. So, if we were to repeat the experiment 100 times, then we expect to have our interval contain the true value 95 times.

2.4.2 PROPERTIES OF MLEs

Maximum likelihood estimation of parameters is a relatively easy proposition for many classes of models that ecologists are interested in. Using standard methods of maximizing the likelihood provides numerically precise parameter estimates, an easy model selection framework based on AIC (discussed in Section 2.6.1) and a number of other generally appealing theoretical asymptotic

properties that we outline here. The term asymptotic means that they hold "as the sample size increases to infinity" which we usually interpret simply as "in large sample sizes." For example, to say that a certain formula gives the asymptotic variance of a parameter estimator means the variance is the correct variance for some hypothetical estimator as the sample size n "approaches infinity," which is the same as saying the error between this expression for variance and the actual variance of the MLE approaches 0 as our sample size increases. This is equivalent to saying that the result is not exact at all, for any specific n, just that we can make the error as small as we like, by increasing n.

One of the greatest things about MLEs, and something that is extremely important from a practical standpoint, is that the "uncertainty" of a MLE can be obtained easily: the variance of the MLE can be well approximated, again in large sample sizes, by the inverse of the *Fisher information* (matrix), which is the matrix of second derivatives of the log-likelihood with respect to the parameter(s) θ, evaluated at the MLE $\widehat{\theta}$. (This matrix of second derivatives is called the *Hessian* matrix). We refer to the inverse of the Fisher information matrix as the asymptotic variance–covariance matrix although typically the term "asymptotic" is omitted in practice and we understand our variances for MLEs as being reasonably valid under "large" sample sizes (without ever declaring or understanding what "large" is). From the asymptotic variance–covariance matrix we obtain the "Wald-type" confidence interval as

$$\widehat{\theta} \pm 1.96\sqrt{\text{Var}(\widehat{\theta})}.$$

where $\text{Var}(\widehat{\theta})$ is the asymptotic variance of $\widehat{\theta}$, taken from the diagonal of the asymptotic variance–covariance matrix. In addition to this simple method for obtaining the variance, MLEs possess the following desirable theoretical properties:

1. *Consistency*: The MLE is asymptotically unbiased. That is, if $\widehat{\theta}(\mathbf{y}_n)$ is an estimator depending on data $\mathbf{y}_n$, then $E_y\{\widehat{\theta}(\mathbf{y}_n)\} \rightarrow \theta$ as $n \rightarrow \infty$.
2. *Asymptotic normality*: The MLE is asymptotically normal with mean θ and variance–covariance matrix equal to the inverse of the Fisher information matrix.
3. *Efficiency*: The MLE achieves the famed Cramér-Rao lower bound as the sample size tends to infinity. What this means is that in a theoretical class of all consistent estimators, no other estimator has a lower asymptotic variance than the MLE.
4. MLEs are *invariant* to reparameterization. That is, if $\widehat{\theta}$ is the MLE of θ then the MLE of a function h of θ, $h(\theta)$, is $h(\widehat{\theta})$.

Basically what all of this means is that we expect to do alright using MLEs as long as n is "large," and furthermore MLEs possess a certain degree of analytic tractability in large sample sizes (we can characterize sampling distributions, compute variances, etc.).

2.4.3 THE DELTA APPROXIMATION

While MLEs are invariant to reparameterization, the variance of MLEs is *not* invariant to transformation. That is, if $\widehat{\theta}$ is the MLE of θ, then the MLE of $h(\theta)$ is $\widehat{h(\theta)} = h(\widehat{\theta})$ but $Var(\widehat{h(\theta)}) \neq h(Var(\widehat{\theta}))$. We often use the delta method of computing variances (Appendix 2 of Cooch and White (2014) is a good

introduction along with technical background and context). Alternatively, we can also use bootstrapping, which we discuss in Section 2.4.3 below.

The delta method goes as follows: Suppose we wish to obtain the variance of a random variable y that is defined by a function of another variable x, denoted by $y = h(x)$ where h is some arbitrary (usually) nonlinear function. We derive a linear approximation to $h(x)$ using a Taylor series expansion of $h(x)$ around its mean $E(x) = \mu$. This produces

$$y = \ h(x) \approx h(\mu) + h'(\mu)(x - \mu)$$

where $h'(\mu)$ is the first derivative of h evaluated at μ. Therefore, because $h(\mu)$ is constant (no longer a function of the random variable x), we have

$$\text{Var}(y) \approx (h'(\mu))^2 \text{Var}(x - \mu)$$

So how do we apply this idea to an estimator of some parameter? We equate x to $\widehat{\theta}$ and $\mu = \text{E}(\widehat{\theta})$ and we assume $\widehat{\theta}$ is unbiased so that $\mu = E(\widehat{\theta}) = \theta$. The variable $y = h(\widehat{\theta})$ is what we seek the variance of in which case we see that

$$\text{Var}(y) \approx (h'(\theta))^2 \text{Var}\left(\widehat{\theta}\right)$$

and because we don't know θ we plug in the estimate we have and find that

$$\text{Var}\left(h\left(\widehat{\theta}\right)\right) \approx \left(h'\left(\widehat{\theta}\right)\right)^2 \text{Var}\left(\widehat{\theta}\right).$$

In ecology we frequently encounter two specific cases: $h(x) = \exp(x)$ and $h(x) = \text{logit}^{-1}(x)$. The former arises in log-linear models such as Poisson regression and the latter of course is the inverse-logit transform for converting linear predictors in logistic regression back to probabilities. Because of their prevalence, we go through both of those cases here. In the case of $h(x) = \exp(x)$ we have $h'(x) = \exp(x)$ and so

$$\text{Var}(\exp(x)) \approx \exp(\mu)^2 \text{Var}(x)$$

and therefore

$$\text{Var}\left(\exp\left(\widehat{\theta}\right)\right) \approx \exp\left(\widehat{\theta}\right)^2 \text{Var}\left(\widehat{\theta}\right)$$

The inverse logit transformation is $h(x) = \exp(x)/(1 + \exp(x))$ and we have to apply the quotient rule in order to differentiate this. We find that

$$h'(x) = \frac{\exp(x)}{(1 + \exp(x))^2}$$

so that

$$\text{Var}\left(\text{logit}^{-1}\left(\widehat{\theta}\right)\right) \approx \frac{\text{Var}\left(\widehat{\theta}\right)\exp\left(\widehat{\theta}\right)^2}{\left(1 + \exp\left(\widehat{\theta}\right)\right)^4}$$

2.4.4 EXAMPLE: CLASSICAL INFERENCE FOR LOGISTIC REGRESSION

We now clarify some of the ideas related to the analysis of specific models using a simple logistic regression model. Let $y_1, y_2, \ldots, y_n$ denote our observations of presence/absence, which we suppose follow a basic logistic regression model, so that the observation model is:

$$y_i \sim Bernoulli(\psi_i)$$

with

$$\text{logit}(\psi_i) = \beta_0 + \beta_1 x_i$$

for sites $i = 1, 2, \ldots, M$ and where x_i is some habitat covariate. And further, we suppose that $y_1, \ldots, y_n$ are mutually *independent*, which as we tried to stress previously is an important part of the probability model specification for determining the joint probability distribution of the sample of n observations. Under independence, we can form the joint distribution as the product of the n constituent Bernoulli components:

$$f(y_1, y_2, \ldots, y_n; \beta_0, \beta_1) = \left\{ \prod_{i=1}^{n} Bernoulli(y_i; \boldsymbol{\beta}) \right\}$$

Here we emphasize the dependence on the parameters $\boldsymbol{\beta}$ by writing this as the argument of the Bernoulli distribution, keeping in mind that there is a logit-linear function relating the canonical parameter ψ and parameters $\boldsymbol{\beta}$. The likelihood is therefore:

$$L(\beta_0, \beta_1; y_1, y_2, \ldots, y_n) \equiv \left\{ \prod_{i=1}^{n} Bernoulli(y_i; \boldsymbol{\beta}) \right\}.$$

The main thing we usually do with the likelihood is maximize it over the parameter space (possible values of $\boldsymbol{\beta}$). The value of $\boldsymbol{\beta}$ that produces the maximum value is the maximum likelihood estimator, which we denote by $\widehat{\boldsymbol{\beta}}$ (we will do this in R shortly). A key point to remember about the likelihood is that it is *not* a probability distribution for the parameters. Although it started out as a probability distribution for y, we have switched the arguments and now think about it as a function of $\boldsymbol{\beta}$.

The following set of R commands will allow us to simulate one data set with a covariate "vegetation height" (`vegHt`):

```
# Simulate a covariate called vegHt for 100 sites
set.seed(2014)  # Set seed so we all get the same values of vegHt
M <- 100        # Number of sites surveyed
vegHt <- runif(M, 1, 3)  # uniform from 1 to 3

# Suppose that occupancy probability increases with vegHt
# The relationship is described by an intercept of -3 and
#    a slope parameter of 2 on the logit scale
beta0 <- -3
beta1 <- 2
psi <- plogis(beta0 + beta1*vegHt) # apply inverse logit
# Now we go to 100 sites and observe presence or absence
z <- rbinom(M, 1, psi)
```

Now, to obtain the MLEs for the parameters $\boldsymbol{\beta}$ we need to maximize the likelihood for the set of observations that we have (or just simulated). The basic strategy for classical estimation based on likelihood that we adopt in this book is to express the likelihood as an R function and then use the standard functions `optim` or `nlm` to maximize it or, equivalently, minimize the negative of the log-likelihood. In the following bit of R code we write out the likelihood function (which we take the logarithm of and negate to return the negative of the log-likelihood) in a long-winded way, by writing the binomial pmf directly but we also show that this can be computed using the `dbinom` function (that line is commented out). Following our definition of the negative log-likelihood here we evaluate the function at different values of the parameter vector; e.g., $\boldsymbol{\beta} = (0, 0)$ and $\boldsymbol{\beta} = (-3, 2)$, and then we use the R function `optim` to obtain the MLEs. For illustration, we also show how the MLEs can be found by a brute-force grid search over the entire parameter space. Finally, we illustrate the use of the R function `glm` for the same, recognizing that the likelihood is that of a logistic GLM:

```
# Definition of negative log-likelihood.
negLogLike <- function(beta, y, x) {
   beta0 <- beta[1]
   beta1 <- beta[2]
   psi <- plogis(beta0 + beta1*x)
   likelihood <- psi^y * (1-psi)^(1-y) # same as next line:
#  likelihood <- dbinom(y, 1, psi)
   return(-sum(log(likelihood)))
}

# Look at (negative) log-likelihood for 2 parameter sets
negLogLike(c(0,0), y=z, x=vegHt)
negLogLike(c(-3,2), y=z, x=vegHt) # Lower is better!

# Let's minimize it formally by function minimisation
starting.values <- c(beta0=0, beta1=0)
opt.out <- optim(starting.values, negLogLike, y=z, x=vegHt, hessian=TRUE)
(mles <- opt.out$par)      # MLEs are pretty close to truth
     beta0      beta1
-2.539793  1.617025

# Alternative 1: Brute-force grid search for MLEs
mat <- as.matrix(expand.grid(seq(-10,10,0.1), seq(-10,10,0.1)))
                                        # above: Can vary resolution (e.g., from 0.1 to 0.01)
nll <- array(NA, dim = nrow(mat))
for (i in 1:nrow(mat)){
   nll[i] <- negLogLike(mat[i,], y = z, x = vegHt)
}
which(nll == min(nll))
mat[which(nll == min(nll)),]

# Produce a likelihood surface, shown in Fig. 2-2.
library(raster)
r <- rasterFromXYZ(data.frame(x = mat[,1], y = mat[,2], z = nll))
mapPalette <- colorRampPalette(rev(c("grey", "yellow", "red")))
```

```
plot(r, col = mapPalette(100), main = "Negative log-likelihood",
xlab = "Intercept (beta0)", ylab = "Slope (beta1)")
contour(r, add = TRUE, levels = seq(50, 2000, 100))

# Alternative 2: Use canned R function glm as a shortcut
(fm <- glm(z ~ vegHt, family = binomial)$coef)

# Add 3 sets of MLEs into plot
# 1. Add MLE from function minimisation
points(mles[1], mles[2], pch = 1, lwd = 2)
abline(mles[2],0)  # Put a line through the Slope value
lines(c(mles[1],mles[1]),c(-10,10))
# 2. Add MLE from grid search
points(mat[which(nll == min(nll)),1], mat[which(nll == min(nll)),2],
pch = 1, lwd = 2)
# 3. Add MLE from glm function
points(fm[1], fm[2], pch = 1, lwd = 2)
```

The likelihood surface is shown in Figure 2.2 along with the maximum value identified (actually, it is the negative log-likelihood). The three different numerical methods of obtaining the estimates yield essentially identical results (the points indicating each estimate are plotted on top of each other) as we expect. You can see how this process could easily be automated to do a simulation study for a particular situation, or even generalized to other types of models (e.g., Poisson regression instead of logistic regression), simply by changing the Bernoulli pmf to a Poisson pmf (i.e., using `dpois` instead of `dbinom`). Of course, in practice, the technical details of what we do are hidden in the guts of functions such as `glm` or `lm` and we hardly ever confront the likelihood directly; see Chapter 3 for more about these functions and the models they specify.

The `hessian=TRUE` option in the call to `optim` produces the Hessian matrix in the returned list `opt.out`, and so we can obtain the asymptotic standard errors (ASE) for the two parameters by doing this:

```
Vc <- solve(opt.out$hessian)    # Get variance-covariance matrix
ASE <- sqrt(diag(Vc))           # Extract asymptotic SEs
print(ASE)

    beta0     beta1
0.8687444 0.4436064

# Compare to SEs reported by glm() function (output thinned)
 summary(glm(z ~ vegHt, family = binomial))

Call:
glm(formula = z ~ vegHt, family = binomial)
Coefficients:
            Estimate Std. Error   z value   Pr(>|z|)
(Intercept)  -2.5397     0.8687    -2.923  0.003462 **
vegHt         1.6171     0.4436     3.645  0.000267 ***
```

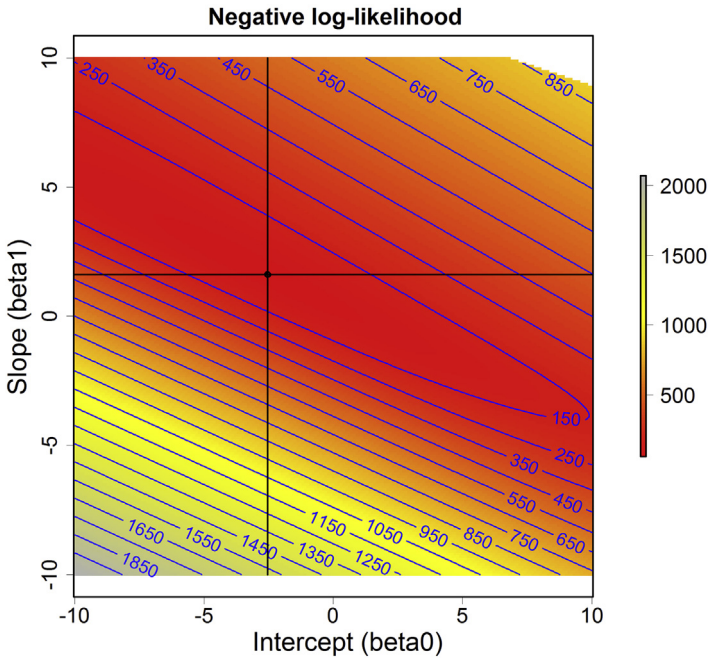

FIGURE 2.2

Likelihood surface for the logistic regression example. The MLE occurs at the point $(\widehat{\beta}_0, \widehat{\beta}_1) = (-2.54, 1.62)$, indicated by a point with intersecting lines through it. The estimates obtained by a grid search and using the `glm` function are also plotted with a point that falls right on top of the MLE.

We see consistency between our ASEs computed by inverting the Hessian directly and those produced by the `glm` function. Therefore, we have faith in our understanding of what `glm` is doing. We summarize some of our likelihood results in a table and produce the figure shown in Figure 2.3:

```
# Make a table with estimates, SEs, and 95% CI
mle.table <- data.frame(Est=mles,
                        ASE = sqrt(diag(solve(opt.out$hessian))))
mle.table$lower <- mle.table$Est - 1.96*mle.table$ASE
mle.table$upper <- mle.table$Est + 1.96*mle.table$ASE
mle.table
             Est       ASE      lower      upper
beta0   2.539793 0.8687444 -4.2425320 -0.8370538
beta1   1.617025 0.4436064  0.7475564  2.4864933

# Plot the actual and estimated response curves
plot(vegHt, z, xlab="Vegetation height", ylab="Occurrence probability")
plot(function(x) plogis(beta0 + beta1 * x), 1.1, 3, add=TRUE, lwd=2)
plot(function(x) plogis(mles[1] + mles[2] * x), 1.1, 3, add=TRUE,
     lwd=2, col="blue")
legend(1.1, 0.9, c("Actual", "Estimate"), col=c("black", "blue"), lty=1,
       lwd=2)
```

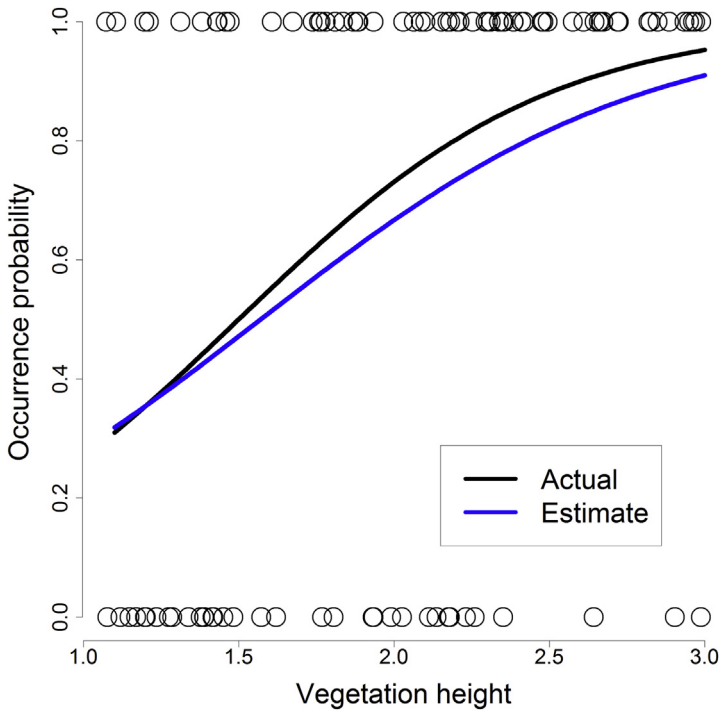

FIGURE 2.3

Actual (black) and fitted (blue) response curve, for expected occurrence $E(z) = \psi$ as a function of covariate "vegetation height" for a simulated data set. Circles show the observed presence (1) and absence (0) data.

2.4.5 BOOTSTRAPPING

For many classical analyses based on likelihood, it is easy enough to obtain variance estimates using the asymptotic results and Fisher Information. However, sometimes we may be worried about the validity of the asymptotic result or else have a really complicated function for which computing derivatives (required for the delta method approximation) is difficult. In such cases, we often compute standard errors using the method of *bootstrapping* (Efron, 1982; Dixon, 2006).

There are two flavors of bootstrapping: parametric and nonparametric. The idea of parametric bootstrapping is that we simulate data sets under the assumed model, but with parameters equal to the MLEs $(\widehat{\theta})$ obtained from our observed data. For each simulated data set, we refit the model and obtain the (new) MLEs for the simulated data set. Each set of MLEs thus obtained is called a bootstrap sample, and together they form the bootstrap distribution that we can summarize to obtain quantities of interest. For example, the variance of the bootstrap samples is an estimate of the sampling variance (the squared standard error). The 0.025 and 0.975 percentiles of the bootstrap distribution form a 95% confidence interval. The difference between the mean of the bootstrap distribution and $\widehat{\theta}$ is an estimate of the bias, and so forth. Nonparametric bootstrapping involves sampling randomly n values from the data set (also with n values) at hand *with replacement*, refitting the model in question, repeating that for

a large number of times and summarizing the MLEs. In this latter case, the explicit parametric assumptions of the model being fitted are no longer relevant to the data generating process, hence the nonparametric label. The R package `unmarked` (Fiske and Chandler, 2011) contains generic parametric and nonparametric bootstrapping methods for certain HMs, including *N*-mixture, distance sampling, occupancy models, and so forth.

Bootstrapping has many purposes, including goodness-of-fit (GoF) evaluation (see Section 2.7), but for now we demonstrate its application to characterizing uncertainty of the MLEs of β_0 and β_1 from our logistic regression example. In addition, we define a parameter being "average occupancy" (`psi.bar`) for which we would like to construct a confidence interval. Otherwise, this quantity would require a delta approximation in order to obtain the ASE of the MLEs.

To carry out the parametric bootstrap for the logistic regression problem we do the following (note that the MLEs were previously computed and the objects `mles` and `mle.table` exist in your workspace):

```
nboot <- 1000   # Obtain 1000 bootstrap samples
boot.out <- matrix(NA, nrow=nboot, ncol=3)
dimnames(boot.out) <- list(NULL, c("beta0", "beta1", "psi.bar"))

for(i in 1:1000){
  # Simulate data
  psi <- plogis(mles[1] + mles[2] * vegHt)
  z <- rbinom(M, 1, psi)

  # Fit model
  tmp <- optim(mles, negLogLike, y=z, x=vegHt, hessian=TRUE)$par
  psi.mean <- plogis(tmp[1] + tmp[2] * mean(vegHt))
  boot.out[i,] <- c(tmp, psi.mean)
}

SE.boot <- sqrt(apply(boot.out, 2, var))  # Get bootstrap SE
names(SE.boot) <- c("beta0", "beta1", "psi.bar")

# 95% bootstrapped confidence intervals
apply(boot.out,2,quantile,c(0.025,0.975))
           beta0     beta1   psi.bar
2.5%   -4.490565 0.8379983 0.5728077
97.5%  -0.978901 2.5974839 0.7828377

# Boostrap SEs
SE.boot
    beta0      beta1     psi.bar
0.89156946 0.45943765 0.05428008

# Compare these with the ASEs for regression parameters
mle.table
             Est       ASE       lower       upper
beta0 -2.539793 0.8687444 -4.2425320 -0.8370538
beta1  1.617025 0.4436064  0.7475564  2.4864933
```

We see that the bootstrapped SEs are slightly larger than the ASEs. This is due to the fact that the bootstrap distribution is slightly skewed; i.e., not normal, as assumed in the calculation of the ASEs.

2.4.6 LIKELIHOOD ANALYSIS OF HMs

2.4.6.1 Discrete Random Effects

We seek now to extend the likelihood estimation framework to HMs, where we will see that a rote application of the basic idea leads us immediately to an insurmountable difficulty. For a case study we consider the occupancy model, our hierarchical extension of the logistic regression model. For this model, the observation model, which previously led directly to the likelihood, is now specified conditional on the random effects (or latent variables) z_i; i.e., it is $f(y_{ij}|z_i, p)$. Clearly it does not make any sense to think about maximizing this because (1) it contains $M + 1$ unknowns—all of the latent variables z_i as well as the parameter p and (2) the parameters of interest, β_0 and β_1 are nowhere to be seen. Therefore, building the likelihood directly from this observation model does not appear useful. Instead, classical analysis of models with random effects is based on the marginal (also called integrated) likelihood, in which we remove the random effects from the conditional likelihood by integration (for the case where the random effects are discrete, we replace our integration with a summation, but the principle is the same).

In general, if z is any random effect, the conditional likelihood is $f(y|z)$, then we would have to compute the marginal distribution of $\mathbf{y}_i$ *unconditional* on the random effect by doing this calculation:

$$f(\mathbf{y}_i|\boldsymbol{\beta}, p) = \int \left\{ \prod_{j=1}^{J} f(y_{ij}|z_i, p) \right\} g(z_i|\boldsymbol{\beta}) dz_i \tag{2.2}$$

where the result, the marginal distribution $f(\mathbf{y}_i|\boldsymbol{\beta}, p)$, conveniently, is no longer a function of z anymore but is a function of the parameters of interest, p and $\boldsymbol{\beta}$. We can now construct our likelihood from these independent marginal pieces $f(\mathbf{y}_i|\boldsymbol{\beta}, p)$ and do standard things to this likelihood, such as obtaining MLEs of parameters p or $\boldsymbol{\beta}$, their asymptotic variances, and so forth.

For the occupancy model, and indeed for many of the HMs that we will consider in this book, it is relatively easy to compute the marginal likelihood because the state variable is discrete, which means the marginal likelihood can be expressed as a summation over what is usually a moderate number of possible values (replacing the integral with a summation in the above expression). To compute the marginal likelihood, we apply the "law of total probability," because z is a discrete random variable (Royle and Dorazio, 2008, p. 34):

$$\Pr(y) = \Pr(y|z = 1)\Pr(z = 1) + \Pr(y|z = 0)\Pr(z = 0)$$

to see that the marginal probability for an observation y_i is:

$$[y_i|p, \psi] = Binomial(y_i; J, p)\psi + I(y_i = 0)(1 - \psi)$$

where we use the notation $I(y_i = 0)$ to be an indicator function that evaluates to 1 if the condition in parentheses holds, and 0 otherwise. The result in this case is the zero-inflated binomial pmf. It is the ordinary binomial, but with additional mass at the value $y = 0$ (Tyre et al., 2003). It is easy to express the marginal likelihood for this HM with binary latent effects using an R function that we provide in the following block of code:

```
set.seed(2014)
M <- 100                              # number of sites
vegHt <- runif(M, 1, 3)               # uniform from 1 to 3
psi <- plogis(beta0 + beta1 * vegHt) # occupancy probability
```

```
z <- rbinom(M, 1, psi)              # realised presence/absence
p <- 0.6                            # detection probability
J <- 3                              # sample each site 3 times
y <-rbinom(M, J, p*z)               # observed detection frequency

# Define negative log-likelihood.
negLogLikeocc <- function(beta, y, x, J) {
    beta0 <- beta[1]
    beta1 <- beta[2]
    p <- plogis(beta[3])
    psi <- plogis(beta0 + beta1*x)
    marg.likelihood <- dbinom(y, J, p)*psi + ifelse(y==0,1,0)*(1-psi)
    return(-sum(log(marg.likelihood)))
}

starting.values <- c(beta0=0, beta1=0,logitp=0)
(opt.out <- optim(starting.values, negLogLikeocc, y=y, x=vegHt,J=J,
                  hessian=TRUE))

$par
     beta0      beta1     logitp
-2.4039229 1.4977667 0.5449382

$value
[1] 122.7039

$counts
function gradient
     170       NA

$convergence
[1] 0

$message
NULL

$hessian
           beta0     beta1     logitp
beta0  17.322109 32.45802   5.268095
beta1  32.458021 65.23173  11.393164
logitp  5.268095 11.39316  34.043000
```

The R function `optim` produces a list with a number of useful things, including the MLE, labeled `par`, the value of the negative log-likelihood (`value`), and the Hessian matrix (`hessian`) that we can invert to obtain the asymptotic variance–covariance matrix or standard errors, as follows:

```
sqrt(diag(solve(opt.out$hessian)))

    beta0     beta1    logitp
0.9258370 0.4799726 0.1770094
```

The function `occu` in the package `unmarked` will fit such occupancy models, which we demonstrate in some detail in Chapter 10.

2.4.6.2 A Continuous Latent Variable

When the latent variable or random effect is discrete, we saw that the marginal likelihood can be computed directly by summing up the likelihood for each possible value of the latent variable and multiplying that by the (prior) probability of each value of the latent variable. When the latent variable (or random effect) is continuous, we have to compute the marginal likelihood explicitly by integration (i.e., Eq. (2.2)). As an example, we consider analysis of a capture–recapture model known as "model M_h," in which individual detection probability contains a normal random effect. The data are the encounter frequencies of individuals over J replicate samples of a population, $y_i \sim Binomial(J, p_i)$ and $\text{logit}(p_i) \sim Normal(\mu, \sigma^2)$. The technical details of model M_h are tangential to the point here, but see Section 6.2 in Royle and Dorazio (2008). For our present purposes the key point is that, in order to compute the marginal likelihood, we have to calculate the integral of the binomial likelihood for each possible encounter frequency (from 0 to J) over the normal prior distribution. In R the commands for doing this integral for each value $j = 0, 1, \ldots, J$ look like this (the following code is not executable as is):

```
marg <- rep(NA, J+1)
for(j in 0:J){
marg[j+1] <- integrate(
   function(x){
      dbinom(j, J, plogis(x)) * dnorm(x, mu, sigma)},
      lower=-Inf,upper=Inf)$value
   }
}
```

The model M_h likelihood also includes a combinatorial term that is a function of unknown population size N ($N = n + n_0$, where n_0 is the number of unobserved individuals), and therefore the parameters of the model include N, μ, and σ^2. We demonstrate the application of this specialized type of HM using a data set on the flat-tailed horned lizard (Figure 2.4; Royle and Young, 2008; Section 6.2 in Royle and Dorazio, 2008) that includes detections of 68 individual lizards over $J = 14$ sample periods.

The R commands that create the data set, define the model M_h likelihood, and obtain the MLEs are as follows:

```
# nx = encounter frequencies, number inds. encountered 1, 2, ..., 14 times
nx <- c(34, 16, 10, 4, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0)
nind <- sum(nx)     # Number of individuals observed
J <- 14             # Number of sample occasions

# Model Mh likelihood
Mhlik <- function(parms){
   mu <- parms[1]
   sigma <- exp(parms[2])
   # n0 = number of UNobserved individuals: N = nind + n0
   n0 <- exp(parms[3])

  # Compute the marginal probabilities for each possible value j=0,..,14
   marg <- rep(NA,J+1)
```

FIGURE 2.4

The flat-tailed horned *lizard* (Phrynosoma mcallii). *(Photo Kevin and April Young.)*

```
    for(j in 0:J){
       marg[j+1] <- integrate(
       function(x){dbinom(j, J, plogis(x)) * dnorm(x, mu, sigma)},
       lower=-Inf,upper=Inf)$value
    }

  # The negative log likelihood involves combinatorial terms computed
  # using lgamma()
  -1*(lgamma(n0 + nind + 1) - lgamma(n0 + 1) + sum(c(n0, nx) * log(marg)))
  }
  (tmp <- nlm(Mhlik, c(-1, 0, log(10)), hessian=TRUE))

  $minimum
  [1] -126.6951

  $estimate
  [1] -2.5648520 -0.2288488  3.6585131

  $gradient
  [1] -1.668833e-05 -1.929834e-05  2.742334e-06
```

```
$hessian
         [,1]      [,2]      [,3]
[1,] 64.58085 25.619056 29.506561
[2,] 25.61906 38.228472 -2.636507
[3,] 29.50656 -2.636507 24.290295

$code
[1] 1

$iterations
[1] 16
```

The likelihood is parameterized in terms of μ, $\log(\sigma)$ and $\log(n_0)$ and therefore we need to back-transform the MLEs to get them on the natural scale. In particular, we care mostly about the population size and so the MLE of n_0 is $\widehat{n}_0 = \exp(3.6585) = 38.8$ and $\widehat{N} = 68 + 38.8 = 106.8$ lizards. We use the delta approximation to obtain the standard error (SE) of $\widehat{N}$ as follows:

```
(SE <- sqrt( (exp(tmp$estimate[3])^2)* diag(solve(tmp$hessian))[3] ) )
[1] 20.80025
```

Which we note is the SE of $\widehat{n}_0$ but, since the number of captured individuals n is fixed (i.e., observed, known), the SE of $\widehat{N}$ is precisely the same as the SE of $\widehat{n}_0$.

2.4.7 THE R PACKAGE `unmarked`

The R package `unmarked` (Fiske and Chandler, 2011) provides a comprehensive platform for analysis of many of the HMs that we cover in this book. It implements likelihood estimation based on marginal likelihood and provides support functions for data organization, summary, and graphical analysis. The package was originally developed by Ian Fiske as a graduate student at North Carolina State University and more or less taken over by Richard Chandler (now at the University of Georgia) in about 2011. The `unmarked` package provides a unified framework for data manipulation, data exploration, model fitting (by maximum likelihood), model selection, model averaging, GoF evaluation, prediction, and it implements ideas of bootstrapping, prediction, empirical Bayes estimation, and other inference and analysis procedures. The core HMs available in `unmarked` are the following:

- Single-season ("static") site-occupancy model (MacKenzie et al., 2002; Tyre et al., 2003)
- Static Royle–Nichols model (Royle and Nichols, 2003)
- Static false-positive occupancy model (Royle and Link, 2006; Miller et al., 2011)
- Static penalized occupancy models (Hutchinson et al., 2015)
- Static binomial *N*-mixture model (Royle, 2004b)
- Static multinomial *N*-mixture model (Royle, 2004a; Dorazio et al., 2005; Langtimm et al., 2011)
- Static hierarchical distance sampling model (Royle et al., 2004)
- "Open-population" versions of many of the above: dynamic occupancy model (MacKenzie et al., 2003); multinomial mixture model with temporary emigration (Chandler et al., 2011); distance sampling with temporary emigration, dynamic *N*-mixture model (Dail and Madsen, 2011), dynamic distance sampling (Sollmann et al., 2015).

Budding hierarchical modelers often have difficulty in figuring out what type of model they should be concerned with for their specific problem. A simple dichotomous key can help guide the user to the appropriate `unmarked` functionality (Figure 2.5). A specific scientific question will often be stated

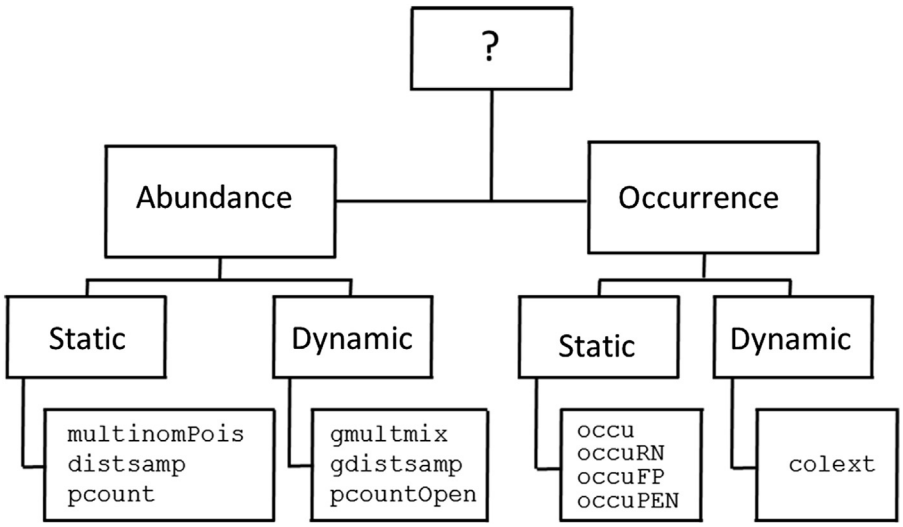

FIGURE 2.5

Decision tree for available models in R package `unmarked`. A scientific or management question (the question mark in the top square) first suggests a focus on either abundance or occurrence; second, the system is static or dynamic; and third, you use a particular sampling method; all of these determine the appropriate model for your study. *(Figure courtesy of R. Chandler.)*

explicitly in terms of abundance or occurrence (question 1), and whether the system is static or dynamic (question 2). Various factors interact to determine a sampling protocol/method (question 3), which then identifies one of the model fitting functions in `unmarked` as illustrated in Figure 2.5.

A unifying element of all HMs handled by `unmarked` is the underlying data structure. They all have "site x replicate" data characterized as repeated samples of sites. In addition, we have site-specific covariates such as "habitat," which only vary by site, or observation-specific covariates (such as "date" or weather conditions), which vary by site and sample occasion (dynamic models may have more complex covariates, such as "annual site-covariates," which we cover in later chapters). The typical data structure is illustrated in Table 2.1.

To use any of the `unmarked` model fitting functions, the data must be put in an object called an `unmarkedFrame` that has a typical implementation as follows:

```
umf <- unmarkedFrame(y = detectionData,
                     siteCovs = siteData,
                     obsCovs = list(wind = windData,
                                    date = dateData))
```

This special class of data frame is necessary because, in the types of HMs we are concerned with, the data structure is not usually amenable to simple matrices, vectors or lists. And, often important metadata such as distance-interval cut-points or measurement units need to be bundled with the data and the `unmarkedFrame` structure facilitates this. Finally, the `unmarkedFrame` structure makes it easy to detect errors in the data organization and allows for custom tools to summarize and visualize data. There is a specific `unmarkedFrame` constructor function for each fitting function, and most of them have the components shown in Table 2.2. Certain `unmarkedFrames` have additional components (Table 2.2).

Table 2.1 Typical data structure for the classes of HMs implemented in the `unmarked` package.

	Detection Data			Site Covariate	Observation Covariate		
	Visit1	Visit2	Visit3	Habitat	Date1	Date2	Date3
Site 1	1	1	1	Good	3	6	10
Site 2	0	0	0	Good	1	7	11
Site 3	1	0	0	Bad	2	9	12
Site 4	0	0	1	Bad	5	6	10

Table 2.2 Basic components of the `unmarkedFrame` data class. Below the horizontal line are optional arguments for certain model classes.

Component	Description
`y`	nSite x nObs matrix of observations (e.g., counts)
`siteCovs`	nSite x nCovs `data.frame` of site-specific covariates
`obsCovs`	A list with nCovs components. Each component is a nSite x nObs `data.frame` of covariate values
`numPrimary`	The number of primary periods (seasons, years, etc.)
`yearlySiteCovs`	A list with nCovs components. Each component is a nSite x nYear matrix of covariate values.
`dist.breaks`	A vector defining the distance intervals for a distance sampling analysis.
`unitsIn`	Measurement units for distance data

A summary of the different model classes, the name of the model fitting function, and the name of the `unmarkedFrame` constructor function is shown in Table 2.3.

The design and structure of `unmarked` accommodates a generic work flow that involves essentially three basic steps:

1. Import data, format it, create an `unmarkedFrame`
2. Fit some models
3. Conduct a summary analysis of the results: Assess model fit, do model selection, make predictions, map abundance, etc..

A conceptual example of this process (not with real data files, but see later chapters) goes like this:
STEP I (import data and create `unmarkedFrame`)

```
pointCountData <- read.csv("myPointCountData.csv")
siteCovariates <- read.csv("mySiteCovariate.csv")
pointCountUMF <- unmarkedFramePCount(y=pointCountData, siteCovs = siteCovariates)
```

STEP II (fit two models)

```
fm1 <- pcount(~1 ~1, data=pointCountUMF)
fm2 <- pcount(~vegHt ~vegHt, data=pointCountUMF)
```

Table 2.3 Model classes in the `unmarked` package and associated functions for fitting the model and creating the `unmarked` data frame.

Model	Fitting Function	unmarkedFrame
Closed populations		
Site-occupancy	`occu`	`unmarkedFrameOccu`
Royle–Nichols	`occuRN`	`unmarkedFrameOccu`
False-positive occupancy	`occuFP`	`unmarkedFrameOccuFP`
Penalized occupancy	`occuPEN`	`unmarkedFrameOccu`
N-mixture	`pcount`	`unmarkedFramePCount`
Removal (multinomial *N*-mix)	`multinomPois`	`unmarkedFrameMPois`
Capture-recapture (multinomial *N*-mix)	`multinomPois`	`unmarkedFrameMpois`
Hierarchical distance sampling	`distsamp`	`unmarkedFrameDS`
Open populations		
Dynamic site-occupancy	`colext`	`unmarkedMultFrame`
Dynamic *N*-mixture	`pcountOpen`	`unmarkedFramePCO`
Multi-scale (temporary-emigration) *N*-mixture	`gpcount`	`unmarkedFrameGPC`
Temporary-emigration removal	`gmultmix`	`unmarkedFrameGMM`
Temporary-emigration capture-recapture	`gmultmix`	`unmarkedFrameGMM`
Temporary-emigration, hierarchical distance sampling	`gdist`	`unmarkedFrameGDS`
Dynamic, hierarchical distance sampling	`gdistOpen`	`unmarkedFrameGDSO`

STEP III (model selection, model fit, prediction, mapping)

```
fms <- fitList(fm1, fm2)
modSel(fms)  # model selection
parboot(fm2) # goodness of fit
predictions <- predict(fm2, type="state", newdata=mapdata)
levelplot(Predicted ~ x.coord+y.coord, data=predictions)
```

We don't dwell further on the capabilities of `unmarked` right now but instead introduce specific aspects of the package as we describe and analyze specific types of HMs in later chapters.

2.5 BAYESIAN INFERENCE

We turn now to a brief introduction to Bayesian inference. As with classical inference based on likelihood, the Bayesian inference process begins with some probability model for the observations that we use to construct the joint distribution of the data. Indeed, there is no difference at all between the joint distribution of the data analyzed by the non-Bayesian who is practicing likelihood inference and that which is analyzed by the Bayesian. The basic probability assumptions that go into creating a model for the observations are exactly the same no matter the inference paradigm being adopted.

Bayesian inference is characterized by two important features: (1) posterior inference and (2) regarding all parameters as realizations of random variables. Posterior inference—which is to say, inference based on the probability distribution of parameters conditional on the data, arises naturally by viewing parameters as being realizations of a random variable. When this is done, we can apply Bayes' rule (see next section) to combine the joint distribution of our observations with the distribution of the parameters (called the prior distribution, see Section 2.5.3), to directly compute the posterior distribution of the parameters in the model. The main conceptual distinction between Bayesian and classical inference, then, is that in Bayesian inference, we use probability directly to characterize uncertainty about parameters whereas in classical inference probability is used to characterize operating properties of statistical procedures. The Bayesian would argue (and so would we, the authors) that, just as it is natural to characterize uncertain outcomes of stochastic processes using probability, it seems natural also to characterize limited information about unknown parameters using probability.

Once this conceptual view of characterizing uncertainty about model parameters using probability is adopted, then inference about *anything* unknown reduces to a simple rote application of a very basic rule of probability, Bayes' rule, which we discuss next.

2.5.1 BAYES' RULE

As its name suggests, Bayesian analysis makes use of Bayes' rule in order to make direct probability statements about model parameters. Given two random variables z and y, Bayes' rule relates the two conditional probability distributions $[z|y]$ and $[y|z]$ by the relationship:

$$[z|y] = \frac{[y|z][z]}{[y]}. \tag{2.3}$$

Generally speaking, these component distributions are characterized as follows: $[y|z]$ is the conditional probability distribution of y given z, $[z]$ is the marginal distribution of z and $[y]$ is the marginal distribution of y. In the context of Bayesian inference we usually associate specific meanings in which $[y|z]$, the observation model, is thought of as "the likelihood," and $[z]$ as the "prior distribution." As such, we can think of this heuristically as a way of updating our prior knowledge about an unknown quantity (expressed by $[z]$) with information obtained from the data ($[y|z]$).

More generally though, Bayes' rule itself is a mathematical fact and there is no debate in the statistical community as to its validity and relevance to many problems. As an example of a simple application of Bayes' rule, we can use it to compute the probability that a site is occupied by a species, even if we have failed to observe it there, based on our canonical HM for species distributions, the occupancy model. Here z denotes species presence ($z = 1$) or absence ($z = 0$), and $\Pr(z = 1) = \psi$ is occupancy or occurrence probability. Let y be the *observed* presence ($y = 1$) or absence ($y = 0$) (or, strictly speaking, detection and nondetection), and let p be the probability that a species is detected in a single survey at a site given that it is present. Thus, $\Pr(y = 1|z = 1) = p$. The interpretation of this is that, if the species is present, we will only observe it with probability p. In addition, we assume here that $\Pr(y = 1|z = 0) = 0$. That is, the species cannot be detected if it is not present, which is a conventional view adopted in most biological sampling problems. If we survey a site J times yielding observations $y_1, \ldots, y_J$ but never detect the species, then this clearly does not imply that the species is absent ($z = 0$). Rather, our degree of belief in $z = 0$ should be made with a probabilistic statement, namely the conditional probability $\Pr(z = 1|y_1 = 0, \ldots, y_J = 0)$. If the J surveys are independent

Bernoulli trails, then the total number of detections is binomial with probability p and sample size J, and we can use Bayes' rule to compute the probability that the species is present given that it is not detected in J samples; i.e., $\Pr(z = 1|y_1 = 0, \ldots, y_J = 0)$. In words, the expression we seek is:

$$\Pr(\text{present}|\text{not detected}) = \frac{\Pr(\text{not detected}|\text{present})\Pr(\text{present})}{\Pr(\text{not detected})}$$

Mathematically, this is

$$\Pr(z = 1|y = 0) = \frac{\Pr(y = 0|z = 1)\Pr(z = 1)}{\Pr(y = 0)} = \frac{(1 - p)^J\psi}{(1 - p)^J\psi + (1 - \psi)}.$$

The denominator here, the probability of not detecting the species, is composed of the probabilities of two distinct events: (1) not observing the species given that it is present (this occurs with probability $(1 - p)^J\psi$) and (2) the species is not present (this occurs with probability $1 - \psi$). To apply this result, suppose that $J = 2$ surveys are done at a wetland for a species of frog, and the species is not detected there. Suppose further that $\psi = 0.8$ and $p = 0.5$ are obtained from a prior study. Then the probability that the species is present at this site, even though it was not detected, is $(1-0.5)^2 \times 0.8/((1-0.5)^2 \times 0.8 + (1 - 0.8)) = 0.5$.

That is, there is a 50/50 chance that the site is occupied despite the fact that the species wasn't observed there. We will see examples of this calculation in Chapter 10 for the occupancy model, and in Chapter 6 in a very similar context to estimate local abundance. In both cases, Bayes' rule forms the basis for estimating the values of the random effects, z and N.

In summary, Bayes' rule provides a simple linkage between the conditional probabilities $[y|z]$ and $[z|y]$, which is useful whenever we need to deduce one from the other.

2.5.2 PRINCIPLES OF BAYESIAN INFERENCE

Bayes' rule as a basic fact of probability is not disputed. What is controversial to some is the scope and manner in which Bayes' rule is applied by Bayesian analysts. Bayesian analysts assert that Bayes' rule is relevant, in general, to all statistical problems by regarding all unknown quantities of a model as realizations of random variables—this includes data (before they are collected), latent variables, missing values that are estimated, predictions, and of course also parameters. Classical (non-Bayesian) analysts sometimes object to regarding parameters as outcomes of random variables, preferring to think of them usually as "fixed but unknown" constants (using the terminology of classical statistics).

By regarding parameters as realizations of some random variable, Bayesians can use Bayes' rule to make direct probability statements about the unknown parameter values. To see the general relevance of Bayes' rule in this context, let y denote observations—i.e., data—and let $[y|\theta]$ be the observation model (often colloquially referred to as the "likelihood"). Suppose θ is a parameter of interest having (prior) probability distribution $[\theta]$. These are combined to obtain the posterior distribution using Bayes' rule:

$$[\theta|y] = \frac{[y|\theta][\theta]}{[y]}$$

The posterior distribution is a direct result of using Bayes' rule to combine the likelihood and the prior distribution. Once we admit the conceptual view of regarding parameters as random variables,

this leads directly to the posterior distribution, a very natural quantity upon which to base inference about things we do not know—including parameters of statistical models. In particular, $[\theta|y]$ is a probability distribution for θ and we can therefore make direct probability statements to characterize uncertainty about θ.

The denominator of Bayes' rule, $[y]$, is the marginal distribution of the data y, which is the numerator with the unknowns eliminated by integration. We note without further explanation right now that, in many practical problems, this integration can be an enormous pain to compute. The main reason that the Bayesian paradigm has become so popular in the last 20 years or so is because methods have been developed for characterizing the posterior distribution that do *not* require that we possess a mathematical understanding of $[y]$, that is, methods that avoid this integration. This means that we never have to compute it or know what it looks like. Thus, while we can understand the conceptual basis of Bayesian inference merely by understanding Bayes' rule—that is really all there is to it—almost the entire rest of Bayesian analysis has to do with numerical methods of characterizing the posterior distribution in order to avoid having to deal with $[y]$. These methods are known collectively as MCMC methods, which we discuss in Section 2.6. But first we turn to a discussion of issues related to prior distributions.

2.5.3 PRIOR DISTRIBUTIONS

The prior distribution $[\theta]$ is an important feature of Bayesian inference. As a conceptual matter, the prior distribution characterizes "prior beliefs" or "prior (or external) information" about a parameter. Indeed, there is considerable disagreement over the generality and function of prior distributions. While it is true that prior distributions may be chosen to reflect subjective "beliefs," this does not often do us much good in practice and we probably want to avoid doing that as much as possible so that we might expect to obtain the same basic results as some other investigator who has the same data. More often, or rather in the vast majority of situations, the prior is chosen to express a lack of specific information, even if previous studies have been done and even if the investigator does in fact know quite a bit about a parameter, in order to "let the data speak" from the current study. Hence, a hardcore Bayesian might complain that most practicing Bayesians are mere "cryptofrequentists": they use the powerful Bayesian computing machinery (see below) to obtain the same old MLEs.

On the other hand, in some circumstances it may be useful to construct prior distributions from previous analyses of data sets conducted from comparable studies (Gopalaswamy et al., 2012). When the different studies are ordered sequentially in time (e.g., across years), the idea of using last year's posterior distribution as this year's prior distribution is eminently sensible. Indeed, in such a sequential analysis, using all data at once in a single analysis, with a flat prior distribution, will in theory lead to exactly the same posterior as when we sequentially use the posterior from the analysis of one part of the data as the prior for the next, until we have used all the data. This may allow one to break apart the analysis of a huge data set, which may perhaps not be possible using BUGS, into smaller subproblems that may be tractable.

A specific case where it might make sense to use an informative prior that is more based on subjective knowledge than previous data is when we have a parameter that is technically not identifiable (or nearly so). We may have a model that makes a good deal of scientific sense, but insufficient data to inform parameters at all levels of the model. In such cases the use of an informative prior is intermediate between estimating a parameter (the ideal case) and fixing it at some specific value,

something that is quite often done in complex models, for instance Leslie-type projection matrix models. Fixing a parameter can be viewed as an informative prior with point mass at a specific value whereas using an informative prior loosens this up a little bit and formally allows for additional uncertainty in the inferences. In sense then, the use of an informative prior is like the use of a fixed value followed by a sensitivity analysis, but all at once.

Despite limited circumstances where using explicit information could be useful, we generally recommend the use of priors that are meant to reflect a lack of information. For example, for probability parameters such as p or ψ from a basic occupancy model, a natural uninformative prior is *Uniform*(0, 1) because it places equal probability on all possible values of p (or all intervals of equal width around any value). For regression coefficients and other parameters that do not have bounded support, we will sometimes use a *Uniform*$(-\infty, \infty)$ prior (also called a "flat" or "improper" prior) when we develop our own MCMC algorithms, but we generally do not use such priors in BUGS. Alternatively, we use what is usually called a diffuse or vague prior, which is some prior (e.g., normal) with a very large variance, or a uniform with suitably wide bounds. Technically, these contain some information, but (ideally) not enough to exert meaningful influence on the posterior.

While we tend to favor priors that express a lack of information, we note that you cannot possibly initiate a study in ecology without having pretty strong ideas of what you are going to observe based on familiarity with the species, related species, and general ecology. Thus it seems like we are not making the most efficient use of the Bayesian paradigm because we are not using this prior information. On the other hand, as noted above, incorporating subjective prior information into a formal analysis is a can of worms we think is better left unopened and that, as a matter of fact, is rarely done in ecology.

At times the situation arises where even what is thought of as being an uninformative prior can inadvertently impose substantial effect on the posterior of a parameter, and that is not desirable. In general, we need to be careful because prior distributions are not invariant to transformation of the parameter, and therefore neither are posterior distributions (see Link and Barker, 2010). Thus a prior that is meant to express a lack of information on one scale, may be very informative on another scale. For example, if we have a flat prior on logit(p) for some probability parameter p, this is very different from having a *Uniform*(0, 1) prior on p. Nonetheless, it is always possible to assess the influence of prior choice by doing a prior sensitivity analysis or by analyzing a model without any data and inspecting the induced priors for estimands other than the basic structural parameters (see Section 5.5.2). It is often the case that the influence of priors is negligible with sufficient data and in a structurally identifiable model.

2.5.4 COMPUTING POSTERIOR DISTRIBUTIONS

Once we have identified the model of interest and specified prior distributions, we need to compute the posterior distribution. In very limited cases, we can identify the posterior distribution analytically. For example, if $y =$ "number of times we detected some species at a sample site," and we assume $y \sim Binomial(J, p)$ with prior distribution $p \sim beta(a, b)$, then it can be shown that the posterior distribution of p given y is also a beta distribution:

$$[p|y] = beta(a + y, b + J - y).$$

The posterior mean is therefore $(a + y)/(a + b + J)$ (you can look this up on Wikipedia). We should recall from our basic statistics class that the MLE of a binomial success parameter is $\widehat{p} = y/J$. Thus, the MLE and the posterior mean are equal if $a = b = 0$, which is in a sense a beta distribution

with effective sample size 0. This is an improper prior distribution (it integrates to ∞), but despite this, the posterior distribution in this case is not unreasonable. It is worth noting also that the beta distribution with parameters $a + y$ and $b + J - y$ has mode $(a + y - 1)/(a + b + J - 2)$. If we use a uniform prior for p (i.e., with parameters $a = b = 1$) then the posterior mode is equivalent to the MLE. This is a general result: if you use a uniform prior then there is a correspondence between the posterior mode and the MLE. We will demonstrate this many times in the book, especially in Chapter 5.

In practice, we cannot usually obtain the posterior distribution analytically. This is because it is not feasible to compute the marginal probability distribution $[y]$, the denominator resulting from application of Bayes' rule. For a long time this impeded the adoption of Bayesian methods by practitioners. The advent of Markov chain Monte Carlo (MCMC) methods has made it easier to characterize posterior distributions of parameters for any model. The power of MCMC is that it allows us to approximate the posterior using simulation without evaluating high dimensional integrals, and to directly sample from the posterior, even when the posterior distribution is not precisely known. The price can be that MCMC may be computationally expensive and most Bayesian analyses take longer to conduct than their corresponding likelihood analyses, where it is possible to do both. Although MCMC first appeared in the scientific literature in 1949 (Metropolis et al., 1953), it was not until the late 1980s that this technique gained widespread use thanks to advances in computational power and speed and important papers including Tanner and Wong (1987), Gelfand and Smith (1990) and others.

2.6 BASIC MARKOV CHAIN MONTE CARLO (MCMC)

Broadly speaking, MCMC (or Markov chain simulation) is a class of methods for drawing random samples (i.e., simulating) from the target posterior distribution (or any distribution for that matter). Thus, even though we might not recognize the posterior as a named distribution or be able to analyze its features analytically (to devise mathematical expressions for the mean and variance such as we did in the beta-binomial problem of the previous section) we can use these MCMC methods to obtain a large sample from the posterior, and then use that sample to characterize features of the posterior using Monte Carlo averages of the simulated values. For example, if we wish to obtain the posterior mean of the parameter p, and we are able to simulate a sample of T values $p^{(1)}, p^{(2)}, \ldots, p^{(T)}$ from the posterior distribution, then the Monte Carlo estimate of the posterior mean of p is the sample mean of the collection of simulated p's:

$$\tilde{\mathrm{E}}(p|\mathbf{y}) = \frac{1}{T}\sum_{t} p^{(t)}.$$

In characterizing posterior quantities in this manner, we induce *Monte Carlo error* (MC error). That is, we expect a difference between the Monte Carlo average and its expected value (the feature it is meant to estimate). This is not the same as *estimation error* in the normal statistical sense because Monte Carlo error is controllable by increasing T and hence, the MC error decreases to 0 as T increases to infinity. The key useful feature of MCMC algorithms is that as T increases to infinity, the distribution of the samples at iteration t converges to the desired posterior distribution.

As a technical matter, MCMC usually produces *dependent* samples from the posterior distribution. The sequence of T values $\theta^{(1)}, \ldots, \theta^{(T)}$ is said to be a Markov chain, which is to say that $\theta^{(t)}$ depends on the preceding value in the chain, so the samples are autocorrelated. This does not have much practical effect on how we go about inference, and we still use Monte Carlo averages and summary statistics, but our effective sample size is less than the length of the MCMC run (T), which is what the effective sample size would be if we could obtain independent samples from the posterior.

The topic of MCMC is far too vast to cover in detail here, although we introduce some general-purpose methods in the following sections and then discuss, in Chapter 5, how to get the various BUGS engines to do your MCMC for you. We think BUGS will be satisfactory for almost all the problems you will need to deal with but, for the other few percent, it helps to be able to write your own MCMC algorithm or to know somebody who can do this for you (and then you have to understand that code). Also, you should always be on the lookout for new MCMC engines for Bayesian model fitting that are more efficient than BUGS.

2.6.1 METROPOLIS–HASTINGS (MH) ALGORITHM

The Metropolis–Hastings (MH) algorithm is a completely generic method for sampling from any distribution, say $[\theta]$. In our applications, $[\theta]$ will typically be the posterior distribution or full conditional distribution (defined below) of θ. The MH algorithm generates candidate values for the parameter(s) of interest from some proposal or "candidate-generating" distribution that may be conditional on the current value of the parameter. This candidate-generating distribution is denoted by $h(\theta^*|\theta^{t-1})$. Here, θ^* is the candidate or proposed value and θ^{t-1} is the value of θ at the previous time step; i.e., at iteration $t-1$ of the MCMC algorithm. The proposed value is accepted with probability

$$r = min\left(1, \frac{[\theta^*]h\left(\theta^{t-1}|\theta^*\right)}{\left[\theta^{t-1}\right]h\left(\theta^*\middle|\theta^{t-1}\right)}\right) \tag{2.4}$$

which is called the MH acceptance probability. The ratio in this expression can sometimes be >1, which is the reason for the $min(1,\cdot)$ construction. This means that if the product of likelihood and prior is greater for the proposed value of θ than for the current value, the former is accepted, while if it is smaller than the proposed value it may still be accepted, but is done so only with a probability equal to the ratio r.

It is useful to note that $h()$ can be any probability distribution, although there are good and bad choices and this candidate generator may or may not depend on the current value of the parameter θ^{t-1}. In practice, we often use a *random walk proposal* and set

$$h\left(\theta^*\middle|\theta^{t-1}\right) = Normal\left(\theta^{t-1}, \delta^2\right)$$

for some tuning parameter δ. This just perturbs the current value of the parameter by some random noise. This particular proposal distribution is symmetric in the sense that $h(a|b) = h(b|a)$. Strictly speaking, when the proposal distribution is symmetric, we speak of a *Metropolis algorithm*, while for asymmetric proposal distributions, we get the MH algorithm (Hastings, 1970). We could as well use something like $Uniform(-B, B)$ for suitably large B as a proposal distribution; this doesn't depend on θ^{t-1}. This is allowable, although we would expect it to produce poor candidate values and therefore low acceptance probabilities and inefficient algorithms.

In the context of using the M/MH algorithms to do MCMC (in which case the target distribution is a posterior distribution), an important fact is, no matter the choice of $h()$, we can compute the MH acceptance probability directly because the marginal distribution of y cancels from both the numerator and denominator of r:

$$r = \frac{[\mathbf{y}|\theta^*]h\left(\theta^{t-1}\middle|\theta^*\right)/[\mathbf{y}]}{\left[\mathbf{y}\middle|\theta^{t-1}\right]h\left(\theta^*\middle|\theta^{t-1}\right)/[\mathbf{y}]} = \frac{[\mathbf{y}|\theta^*]h\left(\theta^{t-1}\middle|\theta^*\right)}{\left[\mathbf{y}\middle|\theta^{t-1}\right]h\left(\theta^*\middle|\theta^{t-1}\right)}$$

This is the magic of the MH algorithm: it obviates the need to evaluate the integrations required to compute the denominator of Bayes' rule. Thus, to use the MH algorithm we only ever have to evaluate

known distributions, those being the likelihood and prior distributions. In addition, we have to be able to simulate from some proposal distribution which can be essentially anything, but in practice is often a simple normal.

In practice, any particular MCMC algorithm needs to run for some number of iterations before the Markov chain "converges" to the target posterior distribution. Samples from this early, transitional period, called the "burn-in" or "warm-up," are discarded and not used in characterizing posterior summaries. This issue is discussed further in Section 2.6.6.

2.6.2 ILLUSTRATION: USING MH FOR A BINOMIAL MODEL

We illustrate the application of the MH algorithm using a simple problem in which we have two observations from a binomial distribution with constant parameter p, and we wish to compute the posterior distribution of p. We will assume a *beta*(a, b) prior for p. For comparison, we will also compute the MLE of p and produce a plot of the likelihood and compare the two. To get things started we simulate some data and define a couple functions that will be needed for these two analyses. In particular, we need the joint distribution of the two observations and we need a function that defines the posterior distribution to within a constant of proportionality (i.e., *not* including the marginal distribution of the data). All of this goes as follows:

```
# Simulate data
set.seed(2016)
y <- rbinom(2, size=10, p = 0.5)

# Define the joint distribution (= likelihood) which we will maximize
jointdis <- function(data, J, p){
   prod(dbinom(data, size=J, p=p))
}

# Posterior is proportional to likelihood times prior
posterior <- function(p, data, J, a, b){
   prod(dbinom(data, size=J, p=p)) * dbeta(p, a, b)
}
```

We require several things in order to set up our MCMC algorithm. First we need a candidate generator or proposal distribution $h()$. For this we decide to use a completely naive and inefficient choice of choosing candidates according to a *Uniform*(0, 1) distribution (we will also consider an alternative shortly). Secondly, we need to pick a starting value for the parameter p. We can pick any value in the interval [0,1] for this and the algorithm will work effectively. For the analysis below we set $p = 0.2$. Then we repeatedly simulate from the candidate generator, compute the MH acceptance probability, and then decide whether to keep or reject the proposed value. Operationally, this final decision is done by flipping a coin with a probability that, in R, is the same as generating a random uniform deviate, say u, and then accepting the value if $u < r$. We execute these various steps for 100,000 iterations and save the current value of p at each iteration:

```
# Do 100,000 MCMC iterations using Metropolis algorithm
# Assume vague prior which is beta(1,1) = Unif(0,1)
mcmc.iters <- 100000
out <- rep(NA, mcmc.iters)
```

```
# Starting value
p <- 0.2

# Begin the MCMC loop
for(i in 1:mcmc.iters){

   # Use a uniform candidate generator (not efficient)
   p.cand <- runif(1, 0, 1)

   # Alternative: random walk proposal
   # p.cand <- rnorm(1, p, 0.05) # Need to reject if > 1 or < 0
   # if(p.cand < 0 | p.cand > 1 )  next

   r <- posterior(p=p.cand, y, J=10, a=1, b=1) / posterior(p=p, y, J=10, a=1, b=1)

   # Generate a uniform r.v. and compare with "r", this imposes the
       correct probability of acceptance
   if(runif(1) < r)
    p <- p.cand

   # Save the current value of p
   out[i] <- p
}
```

The vector `out` contains 100,000 posterior samples of the parameter p. We can use these samples to characterize features of the posterior distribution of p, such as the mean, standard deviation and any percentiles we care to by summarizing the Monte Carlo samples:

```
mean(out)
[1] 0.363855

sd(out)
[1] 0.1004638

quantile(out, c(0.025, 0.975))
     2.5%     97.5%
0.1817478 0.5708719
```

We see that the posterior mean of p is about 0.36, the standard deviation is about 0.10 and 95% of the posterior samples lie between 0.18 and 0.57.

We next evaluate the likelihood at a grid of 200 values of p and plot the likelihood and produce a density plot of the posterior distribution on the same graphics panel (Figure 2.6).

```
# Evaluate likelihood for a grid of values of p
p.grid <- seq(0.1, 0.9, , 200)
likelihood <- rep(NA, 200)

for(i in 1:200){
   likelihood[i] <- jointdis(y, J=10, p=p.grid[i])
}
```

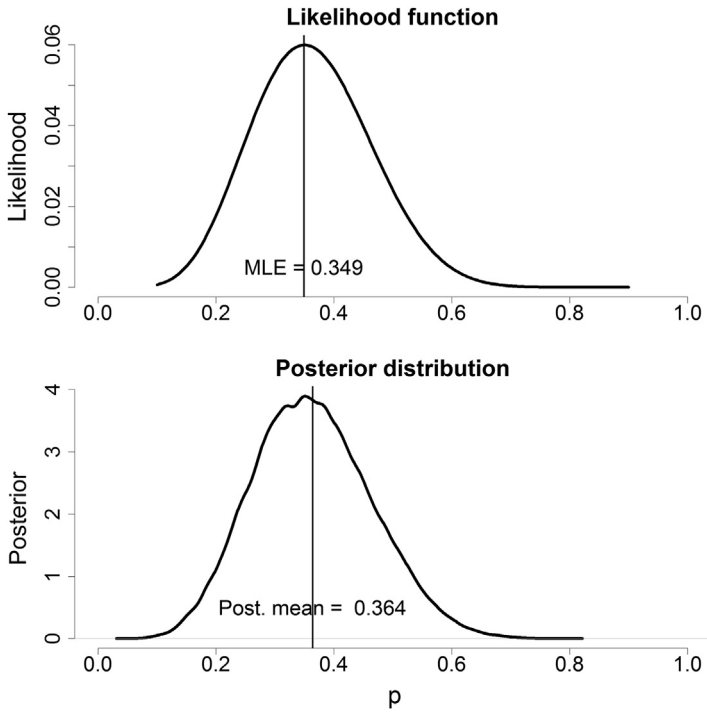

FIGURE 2.6

Likelihood function and posterior distribution for parameter p estimated from a sample of two independent binomial random numbers when vague priors are used in the Bayesian analysis.

```
par(mfrow=c(2,1), mar = c(5,5,3,2))
plot(p.grid, likelihood, xlab="", ylab="Likelihood", xlim=c(0,1), ty = "l", main =
"Likelihood function")
p.hat <- p.grid[likelihood == max(likelihood)]
abline(v = p.hat)
text(p.hat, 0.005, paste("MLE = ", round(p.hat, 3), sep= ""))

plot(density(out), xlim=c(0,1), main = "Posterior distribution", xlab = "p",
ylab = "Posterior")
p.mean <- mean(out)
abline(v = p.mean)
text(p.mean, 0.5, paste("Post. mean = ", round(p.mean, 3),sep=" "))
```

The MLE is $\widehat{p} = 0.349$, which is slightly less than the posterior mean; this is typically the case when the posterior distribution is slightly right-skewed. In this particular instance we expect the mode of the posterior distribution to be precisely equal to the MLE except to within MC error.

2.6.3 METROPOLIS ALGORITHM FOR MULTIPARAMETER MODELS

Normally we have more than one parameter in our model and so we need to devise an MCMC algorithm for handling multiple parameters. How do we do this? In a nutshell, we repeat the single

parameter Metropolis algorithm sequentially for each parameter, but holding each of the other parameters constant. We will demonstrate this approach here using our logistic regression example from Section 2.4.2.

Following our two observation binomial example, we need to make a function that is the unnormalized posterior; i.e., the *numerator* in Bayes rule (likelihood times prior). This time we will define this on the log scale, which will often be necessary for models with many parameters in order to avoid numerical over- or underflow problems. For the logistic regression model we make use of our `negLogLike` function that we used previously for MLE, and we assume $Normal(0, \sigma = 10)$ priors for both parameters. The log-posterior function is, to within an additive constant (that being the marginal distribution of the data):

```
log.posterior <- function(beta0, beta1, z, vegHt){
# Note: "z" and "vegHt" must be input
   loglike <- -1 * negLogLike(c(beta0, beta1), z, vegHt)
   logprior <- dnorm(c(beta0, beta1), 0, 10, log=TRUE)
   return(loglike + logprior[1] + logprior[2])
}
```

Note that, in addition to `negLogLike` from our previous analyses, we also simulated the data `z` and `vegHt` that should already exist in your R workspace (Section 2.4.2).

To get the MCMC algorithm running we need some initial values, which we will generate from a normal distribution, and we need candidate generators ($h()$) that, for each parameter, we will take to be normal distributions with mean being the current value and standard deviation $\delta = 0.3$; i.e.,

$$\beta_0^* \sim Normal(\beta_0, \delta)$$

Note that this normal distribution is *not* part of the model, but rather is a part of the MCMC algorithm which we may modify to improve algorithm performance.

Now we are ready to implement the algorithm, which we do in the following block of code executed for 50,000 iterations:

```
niter <- 50000
out <- matrix(NA, niter, 2, dimnames = list(NULL, c("beta0", "beta1")))

# Initialize parameters
beta0 <- rnorm(1)
beta1 <- rnorm(1)

# Current value of the log(posterior)
logpost.curr <- log.posterior(beta0, beta1, z, vegHt)

# Run MCMC algorithm
for(i in 1:niter){
   if(i %% 1000 == 0)                    # report progress
      cat("iter", i, "\n")
      # Update intercept (beta0)
      # Propose candidate values of beta
```

```
    # If the proposal was not symmetric, would be Metrop-*Hastings*
    beta0.cand <- rnorm(1, beta0, 0.3)    # 0.3 is tuning parameter
    # Evaluate the log(posterior)
    logpost.cand <- log.posterior(beta0.cand, beta1, z, vegHt)
    # Compute Metropolis acceptance probability, r
    r <- exp(logpost.cand - logpost.curr)
    # Keep candidate if it meets criterion (u < r)
    if(runif(1) < r){
       beta0 <- beta0.cand
       logpost.curr <- logpost.cand
    }

    # Update slope (beta1)
    beta1.cand <- rnorm(1, beta1, 0.3)    # 0.3 is tuning parameter
    # Evaluate the log(posterior)
    logpost.cand <- log.posterior(beta0, beta1.cand, z, vegHt)

    # Compute Metropolis acceptance probability
    r <- exp(logpost.cand - logpost.curr)
    # Keep candidate if it meets criterion (u < r)
    if(runif(1) < r){
       beta1 <- beta1.cand
       logpost.curr <- logpost.cand
        }
    out[i,] <- c(beta0, beta1) # Save samples for iteration
  }

# Plot
layout(rbind(c(1,1),
          c(2,2),
          c(3,4)), respect = T) # <- play with these settings

par(oma=c(0,0,0,0),mar=c(5,4,1,1))
plot(out[,1], type="l", xlab="Iteration", ylab="beta0")
plot(out[,2], type="l", xlab="Iteration", ylab="beta1")
plot(density(out[,1]), xlab="beta0", main="")
plot(density(out[,2]), xlab="beta1", main="")
```

The trace plot, or MCMC history, for 50,000 iterations is shown in Figure 2.7 (top two panels). We can produce density plots from these 50,000 posterior samples using the density function, and those are shown in Figure 2.7 (bottom panel).

2.6.4 WHY WE NEED TO USE MCMC FOR LOGISTIC REGRESSION

We will elaborate a little bit on why we do the MCMC analysis of the logistic regression model shown in the previous section instead of just cooking up a formula for the posterior distribution. The logistic regression problem seems so basic and simple that one might think we should easily be able to

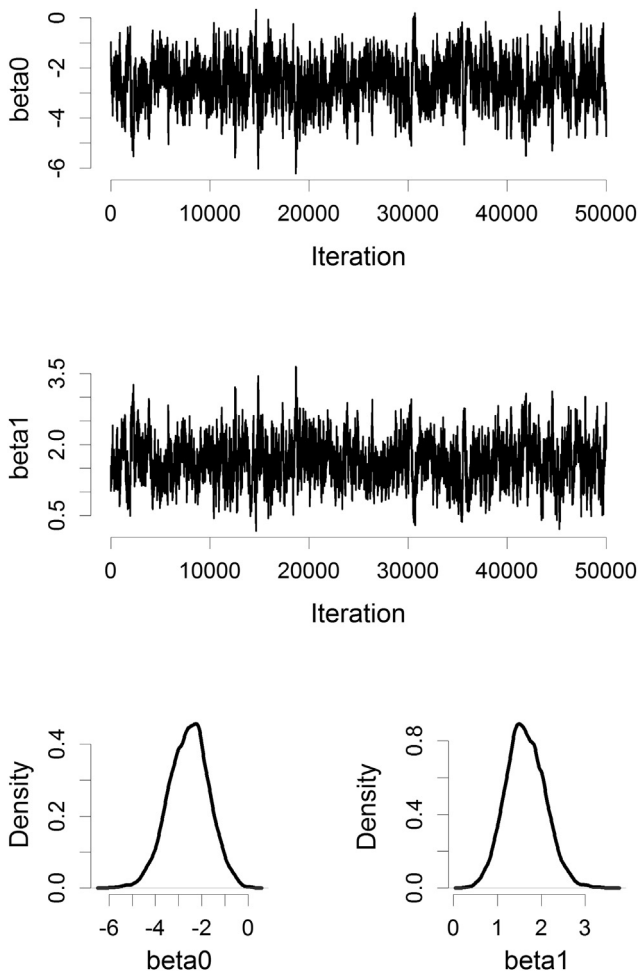

FIGURE 2.7

Output from running an MH algorithm for 50,000 iterations for a logistic regression model: trace plots for the intercept, β_0 (top panel), slope, β_1 (middle panel), and marginal posterior densities (bottom two panels).

compute the posterior distribution directly for this simple two-parameter model! Right? Well, let us see if we can do it!

To carry out a Bayesian analysis, we have to identify the elements needed for Bayes' rule—the "likelihood," the prior distribution for the parameters, and the marginal distribution of the data (the denominator of Bayes' rule). The likelihood is the joint distribution of the observations:

$$f(y_1, y_2, \ldots, y_n|\beta_0, \beta_1) = \left\{ \prod_{i=1}^{n} Bernoulli(\mathbf{y}_i|\boldsymbol{\beta}) \right\}$$

For prior distributions we define $g(\beta_0, \beta_1) \equiv [\beta_0, \beta_1]$ to be the product of two normal distributions each with some variance (we used 100 in the example above but it does not matter here). If our covariate x (`vegHt` in the example) is standardized then this prior should be reasonably flat where the likelihood has mass (sometimes called locally uniform prior) and therefore effectively noninformative. Identifying the joint distribution of the observations and specifying prior distributions allows us to identify the posterior distribution, using Bayes' rule, which is:

$$\pi(\beta_0, \beta_1 | y_1, y_2, \ldots, y_n) = \frac{f(y_1, y_2, \ldots, y_n | \beta_0, \beta_1) g(\beta_0, \beta_1)}{f(y_1, y_2, \ldots, y_n)}. \tag{2.5}$$

While we can easily obtain this conceptual representation of the posterior distribution using Bayes' rule, it will usually *not* be possible to recognize it as a standard named distribution. Therefore, computing posterior summaries such as the mean, or percentiles for a confidence interval, seems to be an elusive problem. The reason for this analytic intractability stems essentially from difficulty in computing the denominator of the Bayes' rule application. In the case of the posterior distribution expressed by Eq. (2.5), the denominator requires that we integrate the numerator over the bivariate prior distribution for β_0 and β_1 which is to say, we have to compute this two-dimensional integral:

$$[\mathbf{y}] = f(y_1, y_2, \ldots, y_n) = \int_{\beta_0} \int_{\beta_1} f(y_1, y_2, \ldots, y_n | \beta_0, \beta_1) g(\beta_0, \beta_1) d\beta_0 d\beta_1.$$

It may be that this can be done analytically (more likely we could do it numerically), and that some further analysis of Eq. (2.5) could be done by some mathematician to produce a simple result that allows us to obtain a formula for the posterior mean, variance, or other summaries. However, for the average analyst, including for the authors, we simply do not know enough math to be able to solve this problem and, importantly, we do not really want to make a big investment in figuring it out because, in the grand scheme of things, we have other more interesting things to do, including about 20 other analyses to finish just in this week alone.

While we cannot do this math (and maybe it cannot even be done, we do not know), we *can* do the MCMC that avoids having to compute this marginal distribution, which we showed in the previous section. In a sense then, MCMC makes Bayesian analysis accessible to the masses who may not be able or interested in solving fun math problems all day.

2.6.5 GIBBS SAMPLING

When we applied MH to multiparameter models we did this by sampling each parameter in succession but holding all other parameters constant, at their current value of the simulation. In our bracket notation, and using the logistic regression as an example, we applied the MH algorithm to sample from the distributions $[\beta_0|\beta_1,\mathbf{y}]$ and $[\beta_1|\beta_0,\mathbf{y}]$. These distributions are called the conditional posterior or "full conditional" distributions because they are the distributions of each parameter conditional on all other variables of the model. Such iterative sampling from full conditional distributions is often called Gibbs sampling (Gelfand et al., 1990). More typically, what we might call "pure" Gibbs sampling is when we can identify the form of each of these full conditional distributions as named distributions and sample from them directly, not using the MH algorithm that we used above. In the cases where we do use the MH algorithm then the resulting algorithm is usually called a

Metropolis-within-Gibbs sampling algorithm. In general, any posterior distribution is completely specified by the set of full conditional distributions, one for *each* unknown quantity of the model (parameters and latent variables).

In most HMs we often have a case where one or more of the full conditionals can be identified as a named distribution, but others cannot. This is because the conditional structure of the model may allow some simplifications due to conditional independencies that are a feature of the model structure. We illustrate this using the occupancy model that we introduced in Section 2.3.1.1. The simple model in which p does not depend on time has the following two elements:

$$y_i \sim Binomial(J, pz_i); \quad i = 1, 2, \ldots, M$$

$$z_i \sim Bernoulli(\psi); \quad i = 1, 2, \ldots, M$$

and the model assumes each y is independent of each other and independent of z_j for $j \neq i$. (Note: We never really say these last two things, but unless a dependency is specified explicitly then we assume independence of variables in a model. This is the only sensible way to interpret written expressions of models.) The Gibbs sampler for this model will have $M + 2$ full conditional distributions: one for each variable z_i and one for each parameter p and ψ. We do not provide all of these here but we make the point that sometimes the full conditional distribution of a parameter has a simplified form by noting that basic probability arguments can be used to show that the full conditional of ψ is the following beta distribution:

$$[\psi|\mathbf{z}] = beta\left(1 + \sum z, 1 + M - \sum z\right)$$

Therefore we can draw samples of ψ directly from the full conditional distribution instead of using a Metropolis algorithm that, while it is very general and avoids having to specify the full conditional distributions, rejects some proposed values of the parameter and hence leads to some inefficiency in characterizing the posterior distribution. The various BUGS engines use some combination of Gibbs sampling from full conditionals, the M/MH algorithm and other more sophisticated MCMC methods.

2.6.6 CONVERGENCE AND MIXING OF MARKOV CHAINS

Once we have carried out an analysis by MCMC, there are many other practical issues that we have to confront. One characteristic of MCMC sampling is that Markov chains take some time to converge to their stationary distribution—in our case the posterior distribution for some parameter given data, $[\theta|y]$. Only when the Markov chain has reached its stationary distribution can the generated samples be used to characterize the posterior distribution. Thus, an important issue we need to address is "have the chains converged?"

Since we do not know what the stationary posterior distribution of our Markov chain should look like, we effectively have no means to assess whether or not it has truly converged to this desired distribution. Most MCMC algorithms only guarantee that, eventually, the samples being generated will be from the target posterior distribution, but no one can tell us how long this will take. There are several things we can do to increase the degree of confidence we have about the convergence of our Markov chains. Some problems are easily detected using simple plots, such as a time-series plot, where parameter values of each MCMC iteration are plotted against the number of iterations. We showed such a trace plot in Figure 2.7.

Typically a period of transience is observed in the early part of the MCMC algorithm, and this is usually discarded as the "burn-in" period (sometimes also called "warm-up"). A quick visual diagnostic to whether convergence has been achieved is that your Markov chains look "grassy" (see Figure 2.7). Another way to check convergence is to update the parameters some more and see if the posterior changes. If the chains have converged to the posterior, the posterior mean, confidence intervals, and other summaries should be relatively static as we continue to run the algorithm. Yet another option is to run several Markov chains and to start them off at different initial values that are overdispersed relative to the posterior distribution. In BUGS we do this with the `n.chains` option. Such multiple chains help to explore different areas of the parameter space simultaneously; if, after a while, all chains oscillate around the same average value, chances are good that they indeed converged to the posterior distribution. Gelman and Rubin (1992) came up with the so-called "R-hat" statistic $(\widehat{R})$ or Brooks–Gelman–Rubin statistic that essentially compares within-chain and between-chain variance to check for convergence of multiple chains (Gelman et al., 2014). The R-hat statistic should be close to 1 if the Markov chains have converged and sufficient posterior samples have been obtained. We demonstrate these ideas with BUGS in later chapters of the book (in particular, see Chapter 5).

2.6.6.1 Slow Mixing and Thinning of Markov Chains

Some models exhibit "poor mixing" of Markov chains, in which case the samples might well be from the posterior (i.e., the Markov chains have converged to the proper stationary distribution) but simply mix or move around the posterior parameter space slowly. Poor mixing can happen for many reasons—when parameters are highly correlated (even confounded) or barely identified from the data, algorithms are inefficient, and probably for other reasons as well.

Slow mixing equates to high autocorrelation in the Markov chain—the successive draws are highly correlated, and thus we need to run the MCMC algorithm much longer to get an effective sample size that is sufficient for estimation, or to reduce the MC error (see below) to a tolerable level. A strategy often used to reduce autocorrelation is "thinning," where only every m^{th} value of the Markov chain output is retained for purposes of summarizing features of the posterior distribution. However, thinning is necessarily inefficient from the stand point of inference—you can always get more precise posterior estimates by using all of the MCMC output regardless of the level of autocorrelation (MacEachern and Berliner, 1994; Link and Eaton, 2012). Practical considerations might necessitate thinning, even though it is statistically inefficient. For example, in models with many parameters or other unknowns, the output files might be enormous and unwieldy to work with. In such cases, thinning is perfectly reasonable. In many cases, how well the Markov chains mix is strongly influenced by parameterization, standardization of covariates, and the prior distributions being used. Some things work better than others, and in Bayesian analyses as in life, the investigator should experiment with different settings and remain calm when things do not work out perfectly on the first attempt.

2.6.6.2 Effective Sample Size and MC Error

The subsequent samples generated from a Markov chain are not independent samples from the posterior distribution, due to the correlation among samples introduced by the Markov process. Thus, sample size has to be adjusted to account for the autocorrelation in subsequent samples (Chapter 8 in Robert and Casella, 2010). This adjusted sample size is referred to as the effective sample size.

Checking the degree of autocorrelation in your Markov chains and estimating the effective sample size your chain has generated should be part of evaluating your model output. This is part of the default output from running the `bugs` function (see Chapter 5) and can be done using the function `effectiveSize` in the `coda` package. If you find that your supposedly long Markov chain has only generated a very short effective sample, you should consider a longer run. The effective sample size determines the accuracy with which you can estimate features of the posterior (such as a 95% credible interval) from your sample.

Another diagnostic useful for assessing the effectiveness of your MCMC analysis is the time-series or MC error—the "noise" introduced into your samples by the stochastic MCMC process. The MC error is printed by default in summaries produced in the WinBUGS GUI. You want the MC error to be smallish relative to the magnitude of the parameter and what smallish means will depend on the purpose of the analysis. For a preliminary analysis you might settle for a few percent whereas for a final analysis then certainly less than 1% is called for. You can run your MCMC algorithm as long as it takes to achieve that (when you have time to wait). A consequence of the MC error is that even for the exact same model, results will usually be slightly different. Thus, as a good rule of thumb, you should avoid reporting MCMC results to more than two or three significant digits!

2.6.7 BAYESIAN ANALYSIS OF HMs

As we discussed in Section 2.4.4, likelihood analysis of HMs is based on the integrated or marginal likelihood, in which we remove the random effects by integration (or summation) from the likelihood that is specified conditional on the random effects. To do a Bayesian analysis of HMs, we apply MCMC directly to the model specified conditional on the random effects. This has the benefit that estimates of the random effects (usually called predictions) are produced directly as part of the MCMC estimation scheme.

For many HMs that we deal with we find that we can sample directly from full conditional distributions for some of the parameters or random effects but perhaps not others. Thus, normally, the MCMC algorithm we devise is a type of hybrid Metropolis-within-Gibbs algorithm. We demonstrate here using the basic occupancy model that we have been simulating from and analyzing in various ways.

The occupancy model: We apply MCMC to the problem of sampling from the joint distribution of our random variables, which now includes a set of random effects, z and therefore we have to include one additional level of full conditional distributions in the MCMC analysis. To sample from the posterior distribution we need to identify the joint distribution of the observations, latent variables and parameters, which has a slightly more complex form than before:

$$f(\mathbf{y}, \mathbf{z}, p, \psi) = \left\{ \prod_{i=1}^{n} \prod_{j=1}^{J} [y_{ij}|z_i, p][z_i|\psi] \right\} [\psi]$$

(Note the latent variables z in here.) However, we still analyze the model by applying MCMC methods in order to obtain a Monte Carlo characterization of posterior summaries. Because of the additional model component for the latent variables, we have an extra full conditional distribution:

$$[z_i|\mathbf{y}, \mathbf{z}_{-i}, p, \psi]$$

for each i. Thus, in total, there are exactly $M + 2$ unknown quantities: the parameters p, and ψ, and M latent variables z_i. Therefore, we need to come up with the $M + 2$ full conditional distributions.

The conditional distribution of z_i given all other unknown parameters in the model, and the data, can be deduced by figuring out $Pr(z_i = 1|y_i, p, \psi)$. Two things need to be said: First, to the right of the vertical line the only data is y_i because the conditional independence assumptions will cause other terms to disappear. Second, we know that if $y_i = 1$ then it must be that $z_i = 1$ and so we only need to figure out this probability for the case $y_i = 0$. In fact, we can compute this probability by a direct application of Bayes' rule:

$$Pr(z_i = 1|y_i = 0, p, \psi) = \frac{Pr(y_i = 0|z_i = 1, p, \psi)Pr(z_i = 1)}{Pr(y_i = 0|z_i = 1)Pr(z_i = 1) + Pr(y_i = 0|z_i = 0)Pr(z_i = 0)}$$

We play around with this for a while and deduce that $z_i|y_i = 0$ is Bernoulli with probability

$$\frac{(1-p)^J\psi}{(1-p)^J\psi + (1-\psi)}.$$

This defines the full conditional for each of the M values of z_i. It remains to define the full conditional for ψ and p. Symbolically, the full conditional for the parameter p has this form:

$$[p|\mathbf{y}, \mathbf{z}, \psi] = \left\{ \prod_i \prod_j [y_{ij}|z_i, p] \right\} [p]$$

This is conditional on each value of z and logically we know that if $z = 0$ then $y = 0$ with probability 1 and so there is no information about p in such cases. Therefore, the full conditional of p is only informed by values of y for which $z = 1$. In fact under a beta prior distribution for p we can show with a little math that

$$[p|\mathbf{y}, \mathbf{z}, \psi] = beta\left(1 + \sum y, 1 + \left(\sum z\right) \times J - \sum y\right).$$

If we did not have covariates in our model that influence occupancy probability then, as we noted in Section 2.5.5, basic arguments can be made that the full conditional of ψ is the following beta distribution:

$$[\psi|\mathbf{z}] = beta\left(1 + \sum z, 1 + M - \sum z\right)$$

However, in keeping with our example in which covariate `vegHt` affects occupancy probability (on the logit scale) we have to update parameters β_0 and β_1. The update of these parameters is done as in Section 2.5.3 for the logistic regression parameters but the "data" for the MH update in this case are the current values of z_i.

Thus, the posterior distribution for this occupancy model is fully characterized by this set of conditional distributions that we iteratively sample from using MCMC algorithms such as the Metropolis-within-Gibbs algorithm. The slightly expanded MCMC algorithm reads as follows (with the output summarized in Figure 2.8:

```
# Simulate the data set
set.seed(2014)
M <- 100                        # number of sites
vegHt <- runif(M, 1, 3)         # uniform from 1 to 3
```

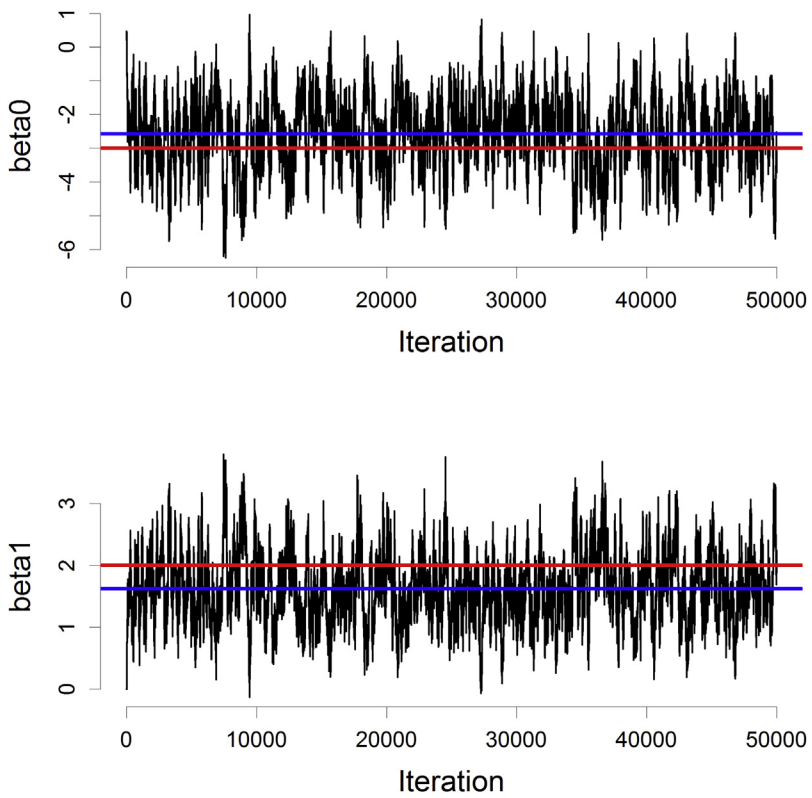

FIGURE 2.8

Output from the MCMC algorithm for site occupancy model using simulated data. Data-generating values (red) are $\beta_0 = -3$ and $\beta_1 = 2$ and posterior means are shown as a blue line. Note the "grassy look" (from the perspective of a vole in a field) indicating suitable mixing of the Markov chains.

```
psi <- plogis(-3 + 2*vegHt)      # occupancy probability
z <- rbinom(M, 1, psi)           # realised presence/absence
p <- 0.6                         # detection probability
J <- 3                           # sample each site 3 times
y <-rbinom(M, J, p*z)            # observed detection frequency

# Number of MCMC iterations to to
niter <- 50000

# Matrix to hold the simulated values
out <- matrix(NA, niter, 3, dimnames = list(NULL, c("beta0", "beta1", "p")))

# Initialize parameters, likelihood, and priors
starting.values <- c(beta0=0, beta1=0)
beta0 <- starting.values[1]
beta1 <- starting.values[2]
z <-ifelse(y>0, 1, 0)
p <- 0.2
```

```
# NOTE: using logistic reg. likelihood function here (defined previously)
loglike <- -1*negLogLike(c(beta0, beta1), z, vegHt)
logprior <- dnorm(c(beta0, beta1), 0, 10, log=TRUE)

# Run MCMC algorithm
for(i in 1:niter) {
   if(i %% 1000 ==0 )  # report progress
   cat("iter", i, "\n")

   # PART 1 of algorithm -- same as before
   # Update intercept (beta0)
   # propose candidate values of beta
   beta0.cand <- rnorm(1, beta0, 0.3) # 0.3 is tuning parameter
   # evaluate likelihood and priors for candidates
   loglike.cand <- -1*negLogLike(c(beta0.cand, beta1), z, vegHt)
   logprior.cand <- dnorm(beta0.cand, 0, 10, log=TRUE)
   # Compute Metropolis acceptance probability (r)
   r <- exp((loglike.cand+logprior.cand) - (loglike + logprior[1]))
   # Keep candidate if it meets the criterion
   if(runif(1) < r){
      beta0 <- beta0.cand
      loglike <- loglike.cand
      logprior[1] <- logprior.cand
   }
   # Update slope (beta1)
   beta1.cand <- rnorm(1, beta1, 0.3) # 0.3 is tuning parameter
   # evaluate likelihood and priors for candidates
   loglike.cand <- -1*negLogLike(c(beta0,beta1.cand), z, vegHt)
   logprior.cand <- dnorm(beta1.cand, 0, 10, log=TRUE)
   # Compute Metropolis acceptance probability r
   r <- exp((loglike.cand+logprior.cand) - (loglike + logprior[2]))
   # Keep the candidates if they meet the criterion
   if(runif(1) < r) {
      beta1 <- beta1.cand
      loglike <- loglike.cand
      logprior[2] <- logprior.cand
   }

   # Part 2 of the algorithm
   # update z. Note we only need to update z if y=0.
   # The full conditional has known form

   psi <- plogis(beta0 + beta1 * vegHt)
   psi.cond <- dbinom(0,J,p) * psi /(dbinom(0, J, p) * psi + (1-psi))
   z[y==0] <- rbinom(sum(y==0), 1, psi.cond[y==0])
   loglike <- -1 * negLogLike(c(beta0, beta1), z, vegHt)
```

```
    # Part 3: update p
   ## The commented code will update p using Metropolis
   ## loglike.p <- sum(log(dbinom(y[z==1],J,p)))
   ## p.cand <- runif(1, 0, 1)
   ## loglike.p.cand <- sum(log(dbinom(y[z==1], J, p.cand)))
   ## if(runif(1) < exp(loglike.p.cand-loglike.p))
   ##     p <- p.cand
   ## This bit draws p directly from its full conditional
    p <- rbeta(1, 1+ sum(y), sum(z)*J +1 - sum(y) )

    # Save MCMC samples
    out[i,] <- c(beta0,beta1,p)
}

# Plot bivariate representation of joint posterior
pairs(out)

# Trace/history plots for each parameter (Fig. 2.8)
op <- par(mfrow=c(2,1))
plot(out[,1], type="l", xlab="Iteration", ylab="beta0")
abline(h = mean(out[,1]), col = "blue", lwd = 2)
abline(h = -3, col = "red", lwd = 2)
plot(out[,2], type="l", xlab="Iteration", ylab="beta1")
abline(h = mean(out[,2]), col = "blue", lwd = 2)
abline(h = 2, col = "red", lwd = 2)
```

2.7 MODEL SELECTION AND AVERAGING

Much of what we talk about in this book has to do with fitting models; i.e., obtaining estimates of parameters for specific models and sampling situations. However, we hardly ever have only a single model that we care to estimate the parameters of. Therefore an important problem that needs to be addressed after we have chosen a class of models and figured out how to fit them is that of choosing among several or many different models within the class, or combining the estimates from several different models ("model averaging"). The idea of model averaging is to average estimates of parameters or predictions over all models under consideration instead of relying on any particular model for making inferences. In this section we briefly address the topics of model selection and averaging under both Bayesian and classical inference paradigms.

2.7.1 MODEL SELECTION BY AIC

Using classical analysis based on likelihood, model selection is easily accomplished using AIC (Burnham and Anderson, 2002). The AIC maximizes "short-term predictive success"; i.e., it can be expected to select models that would best predict a similar data set as the one at hand, similar to

leave-one-out cross-validation (Stone, 1977). The AIC of a model is computed as twice the negative log-likelihood evaluated at the MLE, penalized by the number of parameters (np) in the model:

$$\text{AIC} = -2\,\text{logL}\left(\widehat{\boldsymbol{\beta}}|\mathbf{y}\right) + 2np$$

Models with small values of AIC are preferred. It is common to use AIC with an adjustment for small sample sizes, referred to as AIC_c, which is

$$\text{AIC}_c = -2\,\text{logL}\left(\widehat{\boldsymbol{\beta}}|\mathbf{y}\right) + \frac{2np(np+1)}{n-np-1}$$

where n is the sample size. In applications that involve model selection based on AIC, it is typical to order models by AIC and produce AIC weights, which are exponentiated *differences* between the AIC value (or AIC_c value) for a model and that of the best model. That is, if AIC_m is the AIC value for model m and AIC_{min} is that of the best model, then define $\Delta_m = AIC_m - AIC_{min}$, and the model weight is

$$w_m = \frac{\exp(-(1/2)\Delta_m)}{\sum_m \exp(-(1/2)\Delta_m)}$$

(Burnham and Anderson, 2002, p. 124). The AIC weights are interpreted as model probabilities and are used to produce model-averaged values of parameters. The R package `AICmodavg` (Mazerolle, 2015) does model-averaging of parameters or predictions, for many of the HMs in the R package `unmarked` and other models based on AIC. As we use `unmarked` throughout the book, we will demonstrate application of AIC many times; e.g., see Section 6.9, 7.5.2 and elsewhere.

A big advantage of AIC for model selection is that it is automatic whenever we can compute the marginal likelihood, and it produces weights that can be used directly for model-averaging predictions or parameters that have a consistent interpretation across models. However, one potential problem for applying AIC to HMs, even in some cases when the marginal likelihood can be computed, is that it is not always clear what the effective sample size n should be in the calculation of AIC_c: i.e., is it the number of sites (most relevant for selecting structure for the occupancy part of the model) or the total number of samples (sites $\times$ replicates; most relevant for the detection part of the model).

2.7.2 MODEL SELECTION BY DEVIANCE INFORMATION CRITERION

Spiegelhalter et al. (2002) devised a Bayesian metric for model selection, analogous to AIC (in its application), called the deviance information criterion (DIC), which is a function of model deviance and a measure of effective number of parameters. Model deviance is defined as negative twice the log-likelihood; i.e., for a given model with parameters θ: $\text{Dev}(\theta) = -2\,\text{logL}(\theta|\mathbf{y})$. The DIC is defined as the posterior mean of the deviance, $\overline{\text{Dev}}(\theta)$, plus a measure of model complexity, p_D:

$$\text{DIC} = \overline{\text{Dev}}(\theta) + p_D$$

The standard definition of p_D is

$$p_D = \overline{\text{Dev}}(\theta) - \text{Dev}\left(\overline{\theta}\right)$$

where the second term is the deviance evaluated at the posterior mean of the model parameter(s), $\overline{\theta}$. The p_D here is interpreted as the effective number of parameters in the model. Gelman et al. (2003) suggest a different version of p_D based on one-half the posterior variance of the deviance:

$$P_V = \text{Var}(\text{Dev}(\theta)|\mathbf{y})/2.$$

This is what is produced from WinBUGS and JAGS if they are run using the R packages `R2WinBUGS` or `R2jags` or `jagsUI`, respectively (see Chapter 5). DIC summaries can also be obtained from package `rjags`. The DIC is widely used although its use has been called into question by several authors. Millar (2009) noted that, for HMs, DIC based on the conditional likelihood was invalid (using a case study of zero-inflated count models with overdispersion). Lunn et al. (2013) also noted problems with DIC related to the manner in which p_D is calculated, highlighting two specific issues:

> "1. *p_D is not invariant to reparameterization, in the sense that if the model is rewritten into an equivalent form but in terms of a function $g(\theta)$, then a different p_D may arise since $p(y|\overline{\theta})$ will generally not be equal to $p(y|\overline{g(\theta)})$. This can lead to misleading values in some circumstances and even to negative values of p_D.*
>
> 2. *An inability to calculate p_D when θ contains a categorical parameter, since the posterior mean is not then meaningful. This renders the measure inapplicable to mixture models* [....]."

The latent variables z in an occupancy model, or N in an N-mixture model, are exactly such categorical parameters and thus Royle et al. (2014) were perhaps a bit optimistic in their assessment of DIC when they stated:

> *We think DIC is probably reasonable for certain classes of models that contain only fixed effects, or for which the latent variable structure is the same across models so that only the fixed effects are varied. However, it would be useful to see some calibration of DIC for some standardized model selection problems.*

This does not seem generally true. For example, in capture–recapture models for estimating population size based on data augmentation, different models can suggest different population sizes, and we have observed that this has a strong influence on the model deviance and, in some cases, clearly superior models have much higher posterior deviance. A standard situation is that in which we compare an ordinary spatial capture–recapture model *without* a behavioral response to a model *with* a behavioral response. Even though the behavioral response is clearly important (is large in magnitude), the simpler model will often be favored by the posterior deviance and DIC because it has much lower N and therefore fewer encounter history contributions to the deviance. The use of data augmentation in capture–recapture models is a type of categorical covariate and so this seems consistent with Lunn et al.'s point 2 above.

2.7.3 BAYESIAN MODEL AVERAGING WITH INDICATOR VARIABLES

A convenient way to deal with model selection and averaging problems in Bayesian analysis by MCMC is to use the method of indicator variables (Kuo and Mallick, 1998; see Section 3.4.3 in Royle and Dorazio, 2008; Ntzoufras, 2009; O'Hara and Sillanpää, 2009; Link and Barker, 2010). Using this

approach, we expand the model to include a set of prescribed models as specific reductions of a larger model. To implement the Kuo and Mallick approach, we expand the model to include the latent indicator variables, say w_m, for variable m in the model, such that $w_m = 1$ if the linear predictor contains covariate m, and $w_m = 0$ otherwise. We assume that the indicator variables w_m are mutually independent with

$$w_m \sim Bernoulli(0.5)$$

for each variable $m = 1, 2, \ldots$, in the model. For example, for our simulated example with vegetation height, which only has one covariate, the expanded model for occupancy probability takes this form:

$$\text{logit}(\psi_i) = \beta_0 + \beta_1 w_1 x_i.$$

The posterior probability of the event $w_1 = 1$ is a gauge of the importance of the variable x; i.e., high values of $\Pr(w_1 = 1)$ indicate stronger evidence to support that "x is in the model," whereas values of $\Pr(w_1 = 1)$ close to 0 suggest that x is less important.

In general, using this indicator variable formulation of the model selection problem we can characterize unique models by the sequence of w variables. For just one covariate as above, there are only two models, defined by $w_1 = 1$ or $w_1 = 0$. Considering a more general case; e.g., with three covariates, then each unique sequence (w_1, w_2, w_3) represents a model, and we can tabulate the posterior frequencies of each model by postprocessing the MCMC histories of (w_1, w_2, w_3). This method then produces posterior probabilities for each of the 2^3 models.

This method of indicator variables to do model selection is especially useful for producing model-averaged predictions of latent variables because the MCMC output for a variable represents a posterior sample from all possible models in the set defined by combinations of the w variables. On the other hand, when computing posterior summaries of parameters for specific models, one has to be sure that the MCMC output is subsetted according to the values of the indicator variables w, otherwise the output is a mixture of $w = 1$ and $w = 0$ values, which most of the time is nonsense. See Section 7.6 for detailed applications of the indicator variable approach.

One broader, technical consideration is that posterior model probabilities are well known to be sensitive to priors on parameters (Aitkin, 1991; Link and Barker, 2006). What might normally be viewed as vague priors are not usually innocuous or uninformative when evaluating posterior model probabilities. One solution is to compute posterior model probabilities under a model in which the prior for parameters is fixed at the posterior distribution under the full model (Aitkin, 1991). At a minimum, one should evaluate the sensitivity of posterior model probabilities to different prior specifications (Tenan et al., 2014a).

2.8 ASSESSMENT OF MODEL FIT

Once we decide on a model (or models) and come up with a method of fitting it, it is important to ask the question: Does my model fit the observed data? We define "fit" as follows: Do our data resemble, in some precisely defined manner, realizations from the model? We call the activity of investigating this question the assessment of model fit, or model checking, but a more conventional term is "goodness-of-fit" testing.

Conceptually, we can think of evaluation of model fit as follows: if we simulate data under the model in question, are simulated realizations consistent with ("similar to") the data set that we actually have? Thus in order to formalize the problem of assessing model fit, we need to articulate the manner in which we judge "consistency" in this sense. For either Bayesian or classical inference, the basic strategy to assessing model fit is to come up with a fit statistic that depends on the parameters and the data set, which we denote by $T(\mathbf{y}, \theta)$, and then we compute this for the observed data set, and compare its value with that computed for "perfect" data sets simulated from the correct model.

In the case of classical inference, we will almost always use parametric bootstrapping, in which we simulate data sets using the MLE of the model parameters and then fit the model to each simulated data set. For each simulated data set and fit we compute the value of a fit statistic and compare the value of the fit statistic for observed data with the distribution of that computed from the simulated data sets (this is called the "bootstrap distribution"). The R package `unmarked` (Fiske and Chandler, 2011) contains generic bootstrapping methods. In Bayesian analysis, we adopt the Bayesian *p*-value approach that has a similar feel to the bootstrap, in the sense that we compare the values of a fit statistic for simulated data sets with that computed for the data set at hand.

In either case (classical or Bayesian) we have to come up with one or more fit statistics to use as the basis for the fit assessment. Choice of fit statistic is model and problem-specific and may address only specific features of the model's performance (Gelman et al., 1996). More importantly, any particular fit statistic is unlikely to have good power under all reasonable alternatives and so, in practice, calibration studies may be important, and are needed, but have hardly ever been done for the types of HMs we discuss in this book. At the present time, fit assessment is as much art or intuition as science and we believe it is important and should be more actively investigated.

Perhaps more important than the question of whether a model "fits" is that of *where* does it fit and where does it not. For this, we need to look at patterns in some form of *residual*; i.e., the difference between the observed and the expected data. For HMs, whether in a Bayesian or a non-Bayesian analysis, residuals are not defined in an unambiguous way, but we feel that more can and should be done in this respect. We give some examples in later chapters.

2.8.1 PARAMETRIC BOOTSTRAPPING EXAMPLE

We provide an illustration of using a parametric bootstrapping fit assessment to our occupancy model. We will do this the hard way, by writing our own R code for the problem, although it can also be done easily using the `parboot` function in `unmarked`, which we will demonstrate in subsequent chapters as we introduce `unmarked` capabilities.

The first step in developing the fit assessment is to create a function that simulates data for a given set of parameters. The way this function is set up it always generates the same covariate values but generates a different stochastic response (y):

```
sim.data <- function(beta0 = -3, beta1 = 2, p = 0.6, x=NULL){
# Function allows input of covariate "x", or simulates new

M <- 100
if(is.null(x))
   vegHt <- runif(M, 1, 3) # uniform from 1 to 3
```

```
# Suppose that occupancy probability increases with vegHt
# The relationship is described (default) by an intercept of -3 and
#     a slope parameter of 2 on the logit scale
# plogis is the inverse-logit (constrains us back to the [0-1] scale)
psi <- plogis(beta0 + beta1*vegHt)

# Now we simulated true presence/absence for 100 sites
z <- rbinom(M, 1, psi)

# Now generate observations
J <- 3 # sample each site 3 times
y <- rbinom(M,J,p*z)

list(y=y, J=J, vegHt=vegHt)
}
```

Our function `negLogLikeocc` was defined earlier, but we reproduce it here to maintain the work flow:

```
# This is the negative log-likelihood based on the marginal distribution
# of y. It is the pmf of a zero-inflated binomial random variable.
#
negLogLikeocc <- function(beta, y, x, J) {
   beta0 <- beta[1]
   beta1 <- beta[2]
   p <- plogis(beta[3])
   psi <- plogis(beta0 + beta1*x)
   marg.likelihood <- dbinom(y, J, p) * psi + ifelse(y==0, 1, 0) * (1-psi)
   return(-sum(log(marg.likelihood)))
}
```

The previous analysis of the simulated data is as follows:

```
data <- sim.data()          # Generate a data set

# Let's minimize the negative log-likelihood
starting.values <- c(beta0=0, beta1=0, logitp=0)
opt.out <- optim(starting.values, negLogLikeocc, y=data$y, x=data$vegHt,J=data$J,
hessian=TRUE)
(mles <- opt.out$par)

# Make a table with estimates, SEs, and 95% CI
mle.table <- data.frame(Est=mles,
                        SE = sqrt(diag(solve(opt.out$hessian))))
mle.table$lower <- mle.table$Est - 1.96*mle.table$SE
mle.table$upper <- mle.table$Est + 1.96*mle.table$SE
mle.table
```

To implement the parametric bootstrap GoF analysis, we define a fit statistic as an R function and evaluate that for our data and MLEs. Then, we use the MLEs just obtained to simulate data 100 times and, for each simulated data set, compute the same fit statistic.

```
# Define a fit statistic
fitstat <- function(y, Ey){
   sum((sqrt(y) - sqrt(Ey)))
}
# Compute it for the observed data
T.obs <- fitstat(y, J*plogis(mles[1] + mles[2]*vegHt)*plogis(mles[3]))

# Get bootstrap distribution of fit statistic
T.boot <- rep(NA, 100)
for(i in 1:100){
   # Simulate a new data set and extract the elements. Note we use
   # the previously simulated "vegHt" covariate
   data <- sim.data(beta0=mles[1],beta1=mles[2],p=plogis(mles[3]),x=vegHt)
   # Next we fit the model
   starting.values <- c(0,0,0)
   opt.out <- optim(starting.values, negLogLikeocc, y=data$y, x= data$vegHt, J=data$J,
hessian=TRUE)
   (parms <- opt.out$par)
   # Obtain the fit statistic
   T.boot[i]<- fitstat(y, J*plogis(parms[1] + parms[2]*vegHt)*plogis(parms[3]) )
}
```

What does it all mean? Let us look at the observed value of the statistic and then summary statistics of the bootstrap distribution:

```
(T.obs)
[1] -22.67474

summary(T.boot)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -28.15  -23.27  -21.91  -22.12  -20.85  -17.61
```

We see that our observed statistic T is somewhere between the mean and the first quartile of the bootstrap distribution and thus the observed data set seems consistent with data sets that were simulated under the model. We conclude that the model provides an adequate fit to the data.

2.8.2 BAYESIAN *P*-VALUE

To evaluate GoF in Bayesian analyses, we will typically use the Bayesian p-value as a summary of a posterior predictive check (Gelman et al., 1996). The basic idea is to define a fit statistic (or "discrepancy measure") and compare the posterior distribution of that statistic with the posterior distribution of that statistic for hypothetical perfect data sets for which the model is known to be correct. For example, with count frequency data, a standard measure of fit is based on the *Pearson residuals*:

$$D(y_i, \theta) = \frac{(y_i - \mathrm{E}(y_i))}{\sqrt{\mathrm{Var}(y_i)}}.$$

The fit statistic based on the squared residuals computed from the observations is

$$T(\mathbf{y}, \theta) = \sum_i D(y_i, \theta)^2$$

which can be computed at each iteration of a MCMC algorithm given the current values of parameters that determine the response distribution. At the same time (i.e., at each MCMC iteration), the equivalent statistic is computed for a "new" data set, say $\mathbf{y}^{new}$, simulated using the current parameter values. From the new data set, we compute the fit statistic:

$$T(\mathbf{y}^{new}, \theta) = \sum_i D\left(y_i^{new}, \theta\right)^2$$

and the Bayesian p-value is simply the posterior probability $\Pr(T(\mathbf{y}^{new}) > T(\mathbf{y}))$, which should be close to 0.5 for a good model—one that fits in the sense that the observed data set is consistent with realizations simulated under the model being fitted to the observed data. In practice we judge "close to 0.5" as being "not too close to 0 or 1" which admittedly is somewhat subjective. Another useful fit statistic is the Freeman–Tukey statistic, in which

$$D(\mathbf{y}, \theta) = \sum_i \left(\sqrt{y_i} - \sqrt{\mathrm{E}(y_i)}\right)^2$$

(Brooks et al., 2000), where y_i is the observed value i and $\mathrm{E}(y_i)$ its expected value. In contrast to a Chi-square discrepancy, the Freeman–Tukey statistic obviates the need to pool cells with small expected values and moreover is insensitive to unstable results due to small expected cell frequencies.

In summary, you can see that the Bayesian p-value is easy to compute, and it is widely used as a result. We will later provide several applications of assessing the fit of HMs using Bayesian p-values and parametric bootstrapping, including in Sections 6.7, 7.6 and elsewhere. As with the assessment of model fit using parametric bootstrapping, issues of sensitivity and power of the specific fit statistic and the need for calibration should be considered.

2.9 SUMMARY AND OUTLOOK

HMs represent a series of submodels linked by their conditional dependence structure. There are at least two main reasons for why hierarchical modeling is a sensible thing to do: first, the description of a complex stochastic system is often made much easier by representing it as a series of linked subsystems. This also naturally accommodates variability and uncertainty at all levels of the full system and therefore allows an honest assessment of uncertainty. However, perhaps an even more important reason for adopting HMs than simple practicality is a conceptual, philosophical one: we are convinced that thinking about a system in terms of linked subsystems naturally leads one to a more mechanistic way of statistical model building. HMs naturally enforce a focus on plausible mechanisms underlying the observed data. In this sense, HMs foster the adoption of a "scientific" approach to statistical modeling (Berliner, 1996) exemplified in this book by the clear segregation in the HMs into components describing the latent system and its dynamics and another part describing observation of the system.

There is a sense in which HMs are just glorified GLMs—or rather, a compounding of one GLM with another, or perhaps more than two separate GLMs. We saw that clearly with our two "canonical HMs"—the occupancy model for species distribution, and the *N*-mixture model for species abundance. The occupancy model is the compounding of a logistic regression for observed occupancy state with another logistic regression for true occupancy state, and the *N*-mixture model is a compounding of a logistic regression for observed population size with a Poisson regression for true population size. We will look at many more examples throughout this book including some HMs with three or four levels (our canonical HMs had only two levels each). Owing to the great importance of GLMs for hierarchical modeling, we will spend the entire next chapter on an applied summary of GLMs.

We find that both classical and Bayesian inference paradigms are useful for analysis of HMs. We think you should understand and be able to apply both paradigms because sometimes one has benefits compared with the other. For example, classical inference based on integrated likelihood is sometimes not practically feasible (e.g., for the multispecies occupancy model, Chapter 11). On the other hand, Bayesian analysis using MCMC is completely general and always works, at least in theory. Although, in such cases, MCMC may be very time consuming and lead to Markov chains that mix slowly. On the other hand, analysis of HMs by integrated likelihood provides a convenient and widely accepted framework for model selection based on AIC.

For many chapters of the book we will introduce a class of models (occupancy models, distance sampling, etc..) and then discuss and illustrate likelihood analysis of models using the `unmarked` package whenever we can, stressing consistent work flow and ease of doing standard things like prediction and model selection. We will then do a Bayesian analysis of the class of models using the BUGS language, and then an illustration of a type of model that cannot be done (easily, or in `unmarked`) using likelihood methods. We hope this gives you a sense of the importance of having a pluralistic outlook concerning inference paradigms.

EXERCISES

1. You should be able to apply Bayes' rule to the peregrine falcon example earlier in this chapter to compute the distribution of $X|Y$. That is: how many fledged young are there, given that we have detected birds on 0, 1, 2, … , J visits?
2. For the bootstrap GoF analysis done on the occupancy model, we found that the model appears to fit the data well. Try fitting the wrong model; i.e., without the vegHt covariate, and see if the model *fails* the GoF test.
3. Where we talked about prior distributions and sensitivity we said "if we have a flat prior on logit(p) for some probability parameter p, this is very different from having a *Uniform*(0,1) prior on p." Evaluate this by simulating data for different priors on logit(p) and then back-transforming the simulated values to see what the *implied* prior is for p. Find a normal prior for logit(p) that is approximately uniform on [0,1] for p.
4. Use a Metropolis or MH algorithm to simulate *Normal*(0,1) random variables if all you have access to is a uniform random number generator. Of course you also know the mathematical form for the normal pdf but you do not have a normal random number generator at hand. Verify that your simulated data have the required normal distribution by making a histogram and computing summary statistics.

5. In Section 2.5.2 for a single parameter problem we did the MCMC to obtain the posterior of p and we compared it with the MLE. They appeared slightly different but, because the prior for p is uniform, the posterior mode should be exactly equal (in a mathematical sense) to the MLE. In practice, MC error will make this not so. Repeat the MCMC analysis for many more iterations and see how close the empirical mode of the posterior gets to the MLE. Note that computing the mode is not so easy because there is not a `mode` function in R. So do this in one of two ways: round the output to two or three decimal places and use the `table` function to compute the frequencies of each unique (rounded) value, and the mode is then the most frequent value. Alternatively, use the `density` function and take the value of x that has the highest value of y in the output.