3

GLMS AND BAYESIAN ANALYSIS

A major theme of this book is that spatial capture-recapture models are, for the most part, just generalized linear models (GLMs) wherein the covariate, distance between trap and home range center, is partially or fully unobserved – and therefore regarded as a random effect. Outside of capture-recapture, such models are usually referred to as generalized linear mixed models (GLMMs) and, therefore, SCR models can be thought of as a specialized type of GLMM. Naturally then, we should consider analysis of these slightly simpler models in order to gain some experience and, hopefully, develop a better understanding of spatial capture-recapture models.

In this chapter, we consider classes of GL(M)Ms – Poisson and binomial (i.e., logistic regression) models – that will prove to be enormously useful in the analysis of capture-recapture models of all kinds. Many readers are likely familiar with these models already because they are among the most useful models in ecology and, as such, have received considerable attention in many introductory and advanced texts. We focus on them here in order to introduce the readers to the analysis of such models in **R** and **WinBUGS** or **JAGS**, which we will translate directly to the analysis of SCR models in subsequent chapters.

Bayesian analysis is convenient for analyzing GL(M)Ms because it allows us to work directly with the conditional model – i.e., the model that is conditional on the random effects, using computational methods known as Markov chain Monte Carlo (MCMC). Learning how to do Bayesian analysis of GLMs and GLMMs using the **BUGS** language is, in part, the purpose of this chapter. We focus here on the use of **WinBUGS** because it is the most popular "**BUGS** engine". However, later in the book we transition to another popular **BUGS** engine known as **JAGS** (Plummer, 2009) which stands for *Just Another Gibbs Sampler*. For most of our purposes, the specification of models in either platform is the same, but **JAGS** is under active development at the present time while **WinBUGS** no longer is, having transitioned

to **OpenBUGS** (Lunn et al., 2009) which is still in active development. While we use **BUGS** of one sort or another to do the Bayesian computations, we organize and summarize our data and execute **WinBUGS** or **JAGS** from within **R** using the packages `R2WinBUGS` (Sturtz et al., 2005), `R2jags` (Su and Yajima, 2011) or `rjags` (Plummer, 2009). Kéry (2010), and Kéry and Schaub (2012) provide excellent and accessible introductions to the basics of Bayesian analysis and GL(M)Ms using **WinBUGS**. We don't want to be too redundant with those books and so we avoid a detailed treatment of Bayesian methodology and software usage - instead just providing a cursory overview so that we can move on and attack the problems we're most interested in related to spatial capture-recapture. In addition, there are a number of texts that provide general introductions to Bayesian analysis, MCMC, and their applications in ecology including McCarthy (2007), Kéry (2010), Link and Barker (2010), and King et al. (2008).

While this chapter is about Bayesian analysis of GL(M)Ms, such models are routinely analyzed using likelihood methods too. Later in this book (Chapt. 6), we will use likelihood methods to analyze SCR models but, for now, we concentrate on providing a basic introduction to Bayesian analysis because that is the approach we will use in a majority of cases in later chapters.

3.1 GLMS AND GLMMS

We have asserted already that SCR models work out most of the time to be variations of GL(M)Ms. You might therefore ask: What are these GLM and GLMM models, anyhow? These models are covered extensively in many very good applied statistics books and we refer the reader elsewhere for a detailed introduction. The classical references for GLMs are Nelder and Wedderburn (1972) and McCullagh and Nelder (1989). In addition, we think Kéry (2010), Kéry and Schaub (2012), and Zuur et al. (2009) are all accessible treatments. Here, we'll give the 1 minute treatment of GL(M)Ms, not trying to be complete but rather only to preserve a coherent organization to the book.

The GLM is an extension of standard linear models allowing the response variable to have some distribution from the exponential family of distributions. This includes the normal distribution but also others such as the Poisson, binomial, gamma, exponential, and many more. In addition, GLMs allow the response variable to be related to the predictor variables (i.e., covariates) using a link function, which is usually nonlinear. The GLM consists of three components:

1. A probability distribution for the dependent (or response) variable y, from the exponential family of probability distributions.
2. A "linear predictor" $\eta = \beta_0 + x\beta_1$, where x is a predictor variable (i.e., a covariate).
3. A link function g that relates the expected value of y, $\mathbb{E}(y)$, to the linear predictor, $\mathbb{E}(y) = \mu = g^{-1}(\eta)$. Therefore $g(\mathbb{E}(y)) = \eta = \beta_0 + x\beta_1$.

A key aspect of GLMs is that $g(\mathbb{E}(y))$ is assumed to be a linear function of the predictor variable(s), here x, with unknown parameters, here β_0 and β_1, to be estimated. In standard GLMs, the variance of y is a function V of the mean of y: $\text{Var}(y) = V(\mu)$ (see below for examples). As an example, a Poisson GLM posits that $y \sim \text{Poisson}(\lambda)$ with $\mathbb{E}(y) = \lambda$ and usually the model for the mean is specified using the *log link function* by

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

The variance function is $\text{V}(y_i) = \lambda_i$. To see how a Poisson GLM works, use the **R** code below to simulate some data and then estimate the parameters:

```
> set.seed(13)
> n <- 100            # set sample size
> beta0 <- -2         # set intercept term
> beta1 <- 1.5        # set coefficient
> x <- rnorm(n, 0,1)  # generate a predictor variable, x

> linpred <- beta0 + beta1*x  # calculate linear predictor of E(y)
> y <- rpois(n, exp(linpred))   # generate observations from model
```

The **R** function `glm()` fits a GLM to the data we just generated and returns estimates of β_0 and β_1, which we see are fairly close to the data generating values above:

```
> glm(y ~ 1 + x, family='poisson')      # the fit model
```

This produces the output:

```
Call:  glm(formula = y ~ 1 + x, family = "poisson")

Coefficients:
(Intercept)            x
     -2.007        1.446

[... some output deleted ...]
```

In this summary output, the maximum likelihood estimates (MLEs) of the regression parameters β_0 and β_1 are labeled "`Coefficients`." We see that these are not too different from the data-generating values (-2 and 1.5, respectively).

The binomial GLM posits that $y_i \sim \text{Binomial}(K, p)$ where K is the fixed sample size parameter and $\mathbb{E}(y_i) = K \times p_i$. Usually the model for the mean is specified using the *logit link function* according to

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

Where $\text{logit}(p) = \log(p/(1-p))$. The inverse-logit function, consequently, is $\text{logit}^{-1}(p) = \exp(p)/(1+\exp(p))$.

A GLMM is the extension of GLMs to accommodate "random effects". Often this involves adding a normal random effect to the linear predictor. One simple example is using a random intercept, $\boldsymbol{\alpha}$:

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

where

$$\alpha_i \sim \text{Normal}(\mu, \sigma^2)$$

Many other probability distributions and formulations of the linear predictor might be considered. GLMMs are enormously useful in ecological modeling applications for modeling variation due to subjects, observers, spatial or temporal stratification, clustering, and dependence that arises from any kind of group structure and, of course, because SCR models prove to be a type of GLM with a random effect, but one that does not enter the mean linearly.

3.2 BAYESIAN ANALYSIS

Bayesian analysis is less familiar to many ecological researchers because they are often educated only in the classical statistical paradigm of frequentist inference. But advances in technology and increasing exposure to the benefits of Bayesian analysis are fast making Bayesians out of people or at least making Bayesian analysis an acceptable, general alternative to classical, frequentist inference.

Conceptually, the main thing about Bayesian inference is that it uses probability directly to characterize uncertainty about things we don't know. "Things", in this case, are parameters of models and, just as it is natural to characterize uncertain outcomes of stochastic processes using probability, it seems natural also to characterize information about unknown parameters using probability. At least this seems natural to us and, we think, most ecologists either explicitly adopt that view or tend to fall into that point of view naturally. Conversely, frequentists use probability in many different ways, but never to characterize uncertainty about parameters[1]. Instead, frequentists use probability to characterize the behavior of *procedures* such as estimators or confidence intervals (see below). It is surprising that people readily adopt a philosophy of statistical inference in which the things you don't know (i.e., parameters) should *not* be regarded as random variables, so that, as a consequence, one cannot use probability to characterize one's state of knowledge about them.

3.2.1 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:

$$[z|y] = [y|z][z]/[y]. \tag{3.2.1}$$

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. Generally speaking, these 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 $[y|z]$ is thought of as "the likelihood", $[z]$ as the "prior" and so on. We leave this for later because here the focus is on this expression of Bayes' rule as a basic fact of probability.

[1] To hear this will be shocking to some readers perhaps.

As an example of a simple application of Bayes' rule, consider the problem of determining species presence at a sample location based on imperfect survey information. Let z be a binary random variable that denotes species presence ($z = 1$) or absence ($z = 0$), let $\Pr(z = 1) = \psi$ where ψ is usually called occurrence probability, "occupancy" (MacKenzie et al., 2002) or "prevalence". Let y be the *observed* presence ($y = 1$) or absence ($y = 0$) (or, strictly speaking, detection and non-detection), 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 (but see Royle and Link (2006)). If we survey a site K times but never detect the species, then this clearly does not imply that the species is not present ($z = 0$) at this site but that we failed to observe it. Rather, our degree of belief in $z = 0$ should be made with a probabilistic statement, namely the conditional probability $\Pr(z = 1|y_1 = 0, \ldots, y_K = 0)$. If the K surveys are independent so that we might regard y_k as *iid* Bernoulli trials, then the total number of detections, say y, is Binomial with probability p, and we can use Bayes' rule to compute the probability that the species is present given that it is not detected in K samples, i.e., $\Pr(z = 1|y_1 = 0, \ldots, y_K = 0)$. 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

$$\begin{aligned}\Pr(z = 1|y = 0) &= \frac{\Pr(y = 0|z = 1)\Pr(z = 1)}{\Pr(y = 0)} \\ &= \frac{(1-p)^K\psi}{(1-p)^K\psi + (1-\psi)}.\end{aligned}$$

The denominator here, the probability of not detecting the species, is composed of two parts: (1) not observing the species given that it is present (this occurs with probability $(1-p)^K\psi$) and (2) the species is not present (this occurs with probability $1-\psi$). To apply this result, suppose that $K = 2$ 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.

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.

3.2.2 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, latent variables, and also parameters. Classical (non-Bayesian) analysts sometimes object to regarding parameters as outcomes of random variables. Classically, parameters are thought of as "fixed but unknown" (using the terminology of classical statistics). Indeed, a common misunderstanding on the distinction between Bayesian and frequentist inference goes something like this "in frequentist inference parameters are fixed but unknown but in a Bayesian analysis parameters are random." At best this is a sad caricature of the distinction and at worst it is downright wrong. In Bayesian analysis the parameters are also unknown and, in fact, there is a single data-generating value of each parameter, and so they are also fixed. The difference is that the fixed but unknown values are regarded as having been generated from some probability distribution. Specification of that probability distribution is necessary to carry out Bayesian analysis, but it is not required in classical frequentist inference.

To see the general relevance of Bayes' rule in the context of statistical inference, 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 $[\theta]$ (also simply referred to as the prior). These are combined to obtain the posterior distribution using Bayes' rule, which is:

$$[\theta|y] = [y|\theta][\theta]/[y]$$

Asserting the general relevance of Bayes' rule to all statistical problems, we can conclude that the two main features of Bayesian inference are that: (1) parameters, θ, are regarded as realizations of a random variable and, as a result, (2) inference is based on the probability distribution of the parameters given the data, $[\theta|y]$, which is called the posterior distribution. This is the result of using Bayes' rule to combine the "likelihood" and the prior distribution. The key concept is regarding parameters as realizations of a random variable because, once you admit this conceptual view, this leads directly to the posterior distribution, a very natural quantity upon which to base inference about things we don't know - including parameters of statistical models. In particular, $[\theta|y]$ is a probability distribution for θ and therefore we can make direct probability statements to characterize uncertainty about θ.

The denominator of our invocation of Bayes' rule, $[y]$, is the marginal distribution of the data y. We note without further remark right now that, in many practical problems, this 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 $[y]$. This means we never have to compute it or know what it looks like, or know anything specific about it.

While we can understand the conceptual basis of Bayesian inference merely by understanding Bayes' rule – that's really all there is to it – it is not so easy to understand the basis of classical frequentist inference. What is mostly coherent in frequentist inference is the manner in which procedures are evaluated – the performance of a given procedure is evaluated by "averaging over" hypothetical realizations of y, regarding the *estimator* as a random variable. For example, if $\hat{\theta}$ is an estimator of θ then the frequentist is interested in $\mathbb{E}_y(\hat{\theta}|y)$ which is used to characterize bias. If the expected value of $\hat{\theta}$, when averaged over realizations of y, is equal to θ, then $\hat{\theta}$ is unbiased.

The view of parameters as being random variables allows Bayesians to use probability to make direct probability statements about parameters. Frequentist inference procedures do not permit direct probability statements to be made about parameter values. Instead, the view of parameters as fixed constants and estimators as random variables leads to interpretations that are not so straightforward. For example confidence intervals having the interpretation "95% probability that the interval contains the true value" and p-values being "the probability of observing an outcome of the test statistic as extreme or more than the one observed." These are far from intuitive interpretations to most people. Moreover, this is conceptually problematic to some because we will never get to observe the hypothetical realizations that characterize the performance of our procedure.

While we do tend to favor Bayesian inference for the conceptual simplicity (parameters are random, posterior inference), we mostly advocate for a pragmatic non-partisan approach to inference because, frankly, some of the frequentist methods are actually very convenient in certain situations, and will generally yield very similar inferences about parameters, as we will see in later chapters.

3.2.3 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 information" about a parameter. Indeed, an oft-touted benefit of Bayesian analysis is the ease with which prior information can be included in an analysis. However, more commonly, the prior is chosen to express a lack of prior information, even if previous studies have been done and even if the investigator does in fact know quite a bit about a parameter. This is because the manner in which prior information is embodied in a prior (and the amount of information) is usually very subjective and thus the result can wind up being very contentious; e.g., different investigators might report different results based on subjective assessments of prior information. Thus it is usually better to "let the data speak" and use priors that reflect absence of information beyond the data set being analyzed. An example for an uninformative prior is a Uniform$(0, 1)$ for a probability, or a Uniform(-∞, ∞) (also called a "flat" or "improper" prior) for an unbounded continuous parameter. Alternatively, people use "diffuse priors"; these contain some information, but (ideally) not enough to exert meaningful influence on the posterior. An example for a diffuse prior could be a normal distribution with a large standard deviation.

But still the need occasionally arises to embody prior information or beliefs about a parameter formally into the estimation scheme. In SCR models we often have a parameter that is closely linked to "home range size" and thus auxiliary information on the home range size of a species can be used as prior information, which may improve parameter estimation (e.g., see Chandler and Royle (In press); also Chapt. 18).

At times the situation arises where a prior can inadvertently impose substantial effect on the posterior of a parameter, and that is not desirable. For example, we use data augmentation to deal with the fact that the population size N is an unknown parameter (Royle et al., 2007) which is equivalent to imposing a Binomial(M, ψ) prior on N for some integer M (see Sec. 4.2). One has to take care to make sure that M is sufficiently large so as to not affect the posterior distribution on N (see Fig. 17.6, and also Kéry and Schaub (2012, Ch. 5)). Another situation that we have to be careful of is that prior distributions

are *not* invariant to transformation of the parameter, and therefore neither are posterior distributions (Link and Barker, 2010, Sec. 6.2.1). Thus, a prior that is ostensibly non-informative 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. We show an example where this makes a difference in Chapt. 5. Nonetheless, it is always possible to assess the influence of prior choice, and it is often the case (with sufficient data and a structurally identifiable model) that the influence of priors is negligible.

3.2.4 Posterior inference

In Bayesian inference, we are not focusing on estimating a single point or interval but rather on characterizing a whole distribution – the posterior distribution – from which one can report any summary of interest. A point estimate might be the posterior mean, median, mode, etc.. In many applications in this book, we will compute 95% Bayesian confidence intervals using the 2.5% and 97.5% quantiles of the posterior distribution. For such intervals, it is correct to say $\Pr(L < \theta < U) = 0.95$. That is, "the probability that θ lies between L and U is 0.95".

As an example, suppose we conducted a Bayesian analysis to estimate detection probability (p) of some species at a study site, and we obtained a posterior distribution of beta(20,10) for the parameter p. The following **R** commands demonstrate how we make inferences based upon summaries of the posterior distribution:

```
> post.median <- qbeta(0.5, 20, 10)
[1] 0.6704151

> post.95ci <- qbeta(c(0.025, 0.975), 20, 10)
[1] 0.4916766 0.8206164
```

Thus, we can state that there is a 95% probability that θ lies between 0.49 and 0.82. Fig. 3.1 shows the posterior along with the summary statistics. It is not a subtle thing that such statements cannot be made using frequentist methods, although people tend to say it anyway and not really understand why it is wrong or even that it is wrong.

3.2.5 Small sample inference

The posterior distribution is an exhaustive summary of the state-of-knowledge about an unknown quantity. It is *the* posterior distribution - not an estimate of that thing. It is also not, usually, an approximation except to within Monte Carlo error (in cases where we use simulation to calculate it, see Sec. 3.5.2). One of the great virtues of Bayesian analysis which is not widely appreciated is that posterior inference is not "asymptotic", which is to say, valid in a limiting sense as the sample size tends to infinity. Rather, posterior inference is valid for *any* sample size and, in particular, *the* sample size on-hand. Conversely, almost all frequentist procedures are based on asymptotic approximations to the procedure which is being employed.

There seems to be a prevailing view in statistical ecology that classical likelihood-based procedures are virtuous because of the availability of simple formulas and procedures for

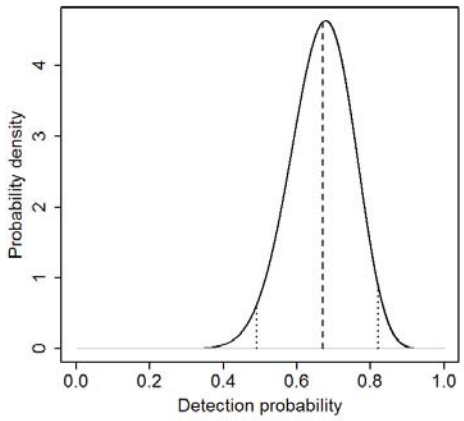

Figure 3.1. Probability density plot of a hypothetical posterior distribution of beta(20,10); dashed lines indicate mean and upper and lower 95% interval

carrying out inference, such as calculating standard errors, doing model selection by Akaike information criterion (AIC), and assessing goodness-of-fit. In large samples, this may be an important practical benefit, but the theoretical validity of these procedures cannot be asserted in most situations involving small samples. This is not a minor issue because it is typical in many wildlife sampling problems – especially in surveys of carnivores or rare/endangered species – to wind up with a small, sometimes extremely small, data set, that is nevertheless extremely valuable (Foster and Harmsen, 2012). For examples: A recent paper (Hawkins and Racey, 2005) on the fossa (*Cryptoprocta ferox*), estimated an adult density of 0.18 adults per sq. km based on a sample size of 20 animals captured over 3 years. Sepúlveda et al. (2007) estimated density of the endangered southern river otter (*Lontra provocax*) based on 12 individuals captured over 3 years, Gardner et al. (2010a) estimated density from a study of the Pampas cat (*Leopardus colocolo*), a species for which very little is known, based on only 22 captured individuals over a two year study period, Trolle and Kéry (2005) reported only 9 individual ocelots captured and Jackson et al. (2006) captured 6 individual snow leopards (*Panthera uncia*) using camera trapping. Thus, almost all likelihood-based analysis of data on rare and/or secretive carnivores necessarily and flagrantly violate one of Le Cam's Basic Principles: "If you need to use asymptotic arguments, do not forget to let your number of observations tend to infinity" (Le Cam, 1990).

The biologist thus faces a dilemma with such data. On one hand, these data sets, and the resulting inference, are often criticized as being poor and unreliable. Or, even worse[2], "the data set is so small, this is a poor analysis." On the other hand, such data

[2]Actual quote from a referee

may be all that is available for species that are extraordinarily important for conservation and management. The Bayesian framework for inference provides a valid, rigorous, and flexible framework that is theoretically justifiable in arbitrary sample sizes. This is not to say that one will obtain precise estimates of density or other parameters, just that your inference is coherent and justifiable from a conceptual and technical statistical point of view. That is, for example when we estimate the density D of some animal population, we report the posterior probability $\Pr(D|data)$ which is easily interpretable and just what it is advertised to be and we don't need to do a simulation study to evaluate how well the reported $\Pr(D|data)$ deviates from the "true" $\Pr(D|data)$ because they are the same quantity.

3.3 CHARACTERIZING POSTERIOR DISTRIBUTIONS BY MCMC SIMULATION

In practice, it is not really feasible to ever compute the marginal probability distribution $[y]$, the denominator resulting from application of Bayes' rule (Eq. 3.2.1). For decades (even centuries!) this impeded the adoption of Bayesian methods by practitioners. Or, the few Bayesian analyses done were based on asymptotic normal approximations to the posterior distribution. While this was useful from a theoretical and technical standpoint and, practically, it allowed people to make the probability statements that they naturally would like to make, it was kind of a bad joke around the Bayesian water-cooler to, on one hand, criticize classical statistics for being, essentially, completely ad hoc in their approach to things but then, on the other hand, have to devise various approximations to what they were trying to characterize. The advent of Markov chain Monte Carlo (MCMC) methods has made it easier to calculate posterior distributions for just about any problem to sufficient levels of precision.

Broadly speaking, MCMC is a class of methods for drawing random samples (i.e., simulating from or just "sampling") from the target posterior distribution. Thus, even though we might not recognize the posterior as a named distribution or be able to analyze its features analytically, e.g., devise mathematical expressions for the mean and variance, 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. What we do with the sample depends on our intentions – typically we obtain the mean or median for use as a point estimate, and take a confidence interval based on Monte Carlo estimates of the quantiles.

3.3.1 What goes on under the MCMC hood

We will develop and apply MCMC methods in some detail for spatial capture-recapture models in Chapt. 17. Here we provide a simple illustration of some basic ideas related to the practice of MCMC.

A type of MCMC method relevant to most problems is Gibbs sampling (Geman and Geman, 1984) which we address in more detail in Chapt. 17. Gibbs sampling involves iterative simulation from the "full conditional" distributions (also called conditional posterior distributions). The full conditional distribution for an unknown quantity is the conditional distribution of that quantity given every other random variable in the model - the data and all other parameters (see Sec. 3.3.2 for rules of how to construct full conditionals).

For example, for a normal regression model [3] with $y \sim \text{Normal}(\beta_0 + \beta_1(x - \bar{x}), \sigma^2)$ where lets say σ^2 is known, the full conditionals are, using "bracket notation",

$$[\beta_0|y, \beta_1]$$

and

$$[\beta_1|y, \beta_0].$$

We might use our knowledge of probability to identify these mathematically. In particular, by Bayes' Rule, $[\beta_0|y, \beta_1] = [y|\beta_0, \beta_1][\beta_0|\beta_1]/[y|\beta_1]$ and similarly for $[\beta_1|y, \beta_0]$. For example, if we have priors for $[\beta_0] = \text{Normal}(\mu_{\beta_0}, \sigma^2_{\beta_0})$ and $[\beta_1] = \text{Normal}(\mu_{\beta_1}, \sigma^2_{\beta_1})$ then some algebra reveals that

$$[\beta_0|y, \beta_1] = \text{Normal}\left(w\bar{y} + (1 - w)\mu_{\beta_0}, (\tau n + \tau_{\beta_0})^{-1}\right) \tag{3.3.1}$$

where $\tau = 1/\sigma^2$ and $\tau_{\beta_0} = 1/\sigma^2_{\beta_0}$ (the inverse of the variance is sometimes called *precision*), and $w = \tau n/(\tau n + \tau_{\beta_0})$. We see in this case that the posterior mean is a *precision-weighted* sum of the sample mean $\bar{y}$ and the prior mean μ_{β_0}, and the posterior *precision* is the sum of the precision of the likelihood and that of the prior. These results are typical of many classes of problems. In particular, note that as the prior precision tends to 0, i.e., $\tau_{\beta_0} \to 0$, then the posterior of β_0 tends to $\text{Normal}(\bar{y}, \sigma^2/n)$. We recognize the variance of this distribution as that of the variance of the sampling distribution of $\bar{y}$ and its mean is in fact the MLE of β_0 for this model. The conditional posterior of β_1 has a very similar form:

$$[\beta_1|y, \beta_0] = \text{Normal}\left(\frac{\tau(\sum_i y_i(x_i - \bar{x})) + \tau_{\beta_1}\mu_{\beta_1}}{\tau \sum_i (x_i - \bar{x})^2 + \tau_{\beta_1}}, (\tau \sum_i (x_i - \bar{x})^2 + \tau_{\beta_1})^{-2}\right) \tag{3.3.2}$$

which might look slightly unfamiliar, but note that if $\tau_{\beta_1} = 0$, then the mean of this distribution is the familiar $\hat{\beta}_1$, and the variance is, in fact, the sampling variance of $\hat{\beta}_1$. The MCMC algorithm for this model has us simulate in succession, repeatedly, from those two distributions. See Gelman et al. (2004) for more examples of Gibbs sampling for the normal model, and we also provide another example in Chapt. 17. A conceptual representation of the MCMC algorithm for this simple model is therefore:

`Algorithm`: Gibbs Sampling for linear regression

`0. Initialize` β_0 `and` β_1

`Repeat` {

`1. Draw a new value of` β_0 `from Eq.` 3.3.1

`2. Draw a new value of` β_1 `from Eq.` 3.3.2

}

As we just saw for this simple "normal-normal" model, it is sometimes possible to specify the full conditional distributions analytically. In general, when certain so-called conjugate prior distributions are used, which have an analytic form that, in a statistical

[3] We center the independent variable here so that things look more familiar in the result

sense, "matches" the likelihood, then the form of the full conditional distributions is also similar to that of the observation model. In this normal-normal case, the normal distribution for the mean parameters is the conjugate prior for the normal observation model, and thus the full-conditional distributions are also normal. This is convenient because, in such cases, we can simulate directly from them using standard methods (or **R** functions). But, in practice, we don't really ever need to know such things because most of the time we can get by using a simple algorithm, called the Metropolis-Hastings (henceforth "MH") algorithm, to obtain samples from these full conditional distributions without having to recognize them as specific, named, distributions. This gives us enormous freedom in developing models and analyzing them without having to resolve them mathematically because to implement the MH algorithm we need only identify the full conditional distribution up to a constant of proportionality, that being the marginal distribution in the denominator (e.g., $[y|\beta_1]$ above).

We will talk about the Metropolis-Hastings algorithm shortly, and we will use it extensively in the analysis of SCR models (e.g., Chapt. 17).

3.3.2 Rules for constructing full conditional distributions

The basic strategy for constructing full-conditional distributions for devising MCMC algorithms can be reduced conceptually to a couple of basic steps summarized as follows:

(step 1) Identify all stochastic components of the model and collect their probability distributions;
(step 2) Express the full conditional in question as proportional to the product of all probability distributions identified in step 1;
(step 3) Remove the ones that don't have the focal parameter in them.
(step 4) Do some algebra on the result in order to identify the resulting probability distribution function (pdf) or mass function (pmf).

Of the 4 steps, the last of those is the main step that requires quite a bit of statistical experience and intuition because various algebraic tricks can be used to reshape the mess into something recognizable – i.e., a standard, named distribution. But step 4 is not necessary if we decide instead to use the Metropolis-Hastings algorithm as described below.

In the context of our simple linear regression model that we've been working with, to characterize $[\beta_0|y, \beta_1]$ we first apply step 1 and identify the model components as: $[y|\beta_0, \beta_1]$, with prior distributions $[\beta_0]$ and $[\beta_1]$. Step 2 has us write $[\beta_0|y, \beta_1] \propto [y|\beta_0, \beta_1][\beta_0][\beta_1]$. Step 3: We note that $[\beta_1]$ is not a function of β_0 and therefore we remove it to obtain $[\beta_0|y, \beta_1] \propto [y|\beta_0, \beta_1][\beta_0]$. Similarly, applying step 2 and 3 for β_1 we obtain $[\beta_1|y, \beta_0] \propto [y|\beta_0, \beta_1][\beta_1]$. We apply step 4 and manipulate these algebraically to arrive at the result (which we provided in Eqs. 3.3.1 and 3.3.2) or, alternatively, we can sample them indirectly using the Metropolis-Hastings algorithm, which we discuss now.

3.3.3 Metropolis-Hastings 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 full conditional distribution of θ. While we sometimes use Gibbs sampling, we seldom use "pure" Gibbs

sampling because full conditionals do not always take the form of known distributions we can sample from directly. In such cases, we use MH to sample from the full conditional distributions. When the MH algorithm is used to sample from full conditional distributions of a Gibbs sampler the resulting hybrid algorithm is called *Metropolis-within-Gibbs*. In Sec. 3.6.3 we will construct such an algorithm for a simple class of models. We discuss both the Gibbs and the MH algorithm, as well as their hybrid in more depth in Chapt. 17.

The MH algorithm generates candidate values for the parameter(s) we want to estimate from some proposal or candidate-generating distribution that may be conditional on the current value of the parameter, 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

$$r = \frac{[\theta^*]h(\theta^{t-1}|\theta^*)}{[\theta^{t-1}]h(\theta^*|\theta^{t-1})}$$

which is called the MH acceptance probability. This ratio can sometimes be > 1 in which case we set it equal to 1. It is useful to note that $h()$ can be any probability distribution.

In the context of using the MH algorithm to do MCMC (in which case the target distribution is a full-conditional or 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. This is the magic of the MH algorithm.

3.4 BAYESIAN ANALYSIS USING THE BUGS LANGUAGE

We won't be too concerned with devising our own MCMC algorithms for every analysis, although we will do that a few times for fun. More often, we will rely on the freely available software package **WinBUGS** or **JAGS** for doing this. We will always execute these **BUGS** engines from within **R** using the `R2WinBUGS` (Sturtz et al., 2005) or, for **JAGS**, the `R2jags` (Su and Yajima, 2011) or `rjags` (Plummer, 2009) packages. **WinBUGS** and **JAGS** are MCMC black boxes that take a pseudo-code description (i.e., written in the **BUGS** language) of all of the relevant stochastic and deterministic elements of a model and generate an MCMC algorithm for that model. But you never get to see the algorithm. Instead, **WinBUGS/JAGS** will run the algorithm and return the Markov chain output - the posterior samples of model parameters.

The great thing about using the **BUGS** language is that it forces you to become intimate with your statistical model - you have to write each element of the model down, admit (explicitly) all of the various assumptions, understand what the actual probability assumptions are and how data relate to latent variables and data and latent variables relate to parameters, and how parameters relate to one another.

While we normally use **WinBUGS**, we note that **OpenBUGS** is the current active development tree of the **BUGS** project. See Kéry (2010) and Kéry and Schaub (2012, especially Appendix 1) for more on practical analysis in **WinBUGS**. Those books should be consulted for a more comprehensive introduction to using **WinBUGS**. Recently we have migrated many of our analyses to **JAGS** (Plummer, 2009), which we adopt later in

the book. You can refer to Hobbs (2011) for an ecological introduction to **JAGS**. Next, we provide an example of a Bayesian analysis using **WinBUGS**.

3.4.1 Linear regression in WinBUGS

We provide a brief introductory example of a normal regression model using a small simulated data set. The following commands are executed from within your **R** workspace. First, simulate a covariate x and observations y having prescribed intercept, slope and variance:

```
 > x <- rnorm(10)
 > mu <- -3.2 + 1.5*x
 > y <- rnorm(10,mu,sd=4)
```

The **BUGS** model specification for a normal regression model is written within **R** as a character string input to the command `cat()` and then dumped to a text file named `normal.txt`:

```
> cat("
 model{
   for (i in 1:10){
      y[i] ~ dnorm(mu[i],tau)          # the likelihood
      mu[i] <- beta0 + beta1*x[i]      # the linear predictor
    }
   beta0 ~ dnorm(0,.01)                # prior distributions
   beta1 ~ dnorm(0,.01)
   sigma ~ dunif(0,100)
   tau <- 1/(sigma*sigma)              # tau is the precision
}                                      #   and a derived parameter
",file="normal.txt")
```

Alternatively, you can write the model specifications directly within a text file and save it in your current working directory, but we do not usually take that approach in this book.

The **BUGS** dialects[4] parameterize the normal distribution in terms of the mean and inverse-variance, called the precision. Thus, `dnorm(0,.01)` implies a variance of 100. We typically use diffuse normal priors for mean parameters, β_0 and β_1 in this case, but sometimes we might use uniform priors with suitable bounds $-B$ and $+B$. Also, we typically use a Uniform$(0, B)$ prior on standard deviation parameters (Gelman, 2006). But sometimes we might use a gamma prior on the precision parameter τ. In a **BUGS** model file, every variable referenced in the model description has to be either data, which will be input (see below), a random variable which must have a probability distribution associated with it using the tilde character "$\sim$" (a.k.a. "twiddle") or it has to be a derived parameter connected to variables and data using an assignment arrow: "`<-`".

To fit the model, we need to describe various data objects to **WinBUGS**. In particular, we create an **R** list object called `data` which are the data objects identified in the **BUGS** model file. In the example, the data consist of two objects which exist as y and x in the

[4]We use this to mean **WinBUGS**, **OpenBUGS** and **JAGS**

R workspace and also in the **WinBUGS** model definition. We also create an **R** function that produces a list of starting values, `inits`, that get sent to **WinBUGS**. In general, starting values are optional. We recommend to always provide reasonable starting values where possible, both for structural parameters and also random effects[5]. Finally, we identify the names of the parameters (labeled correspondingly in the **WinBUGS** model specification) that we want **WinBUGS** to save the MCMC output for. In this example, we will "monitor" the parameters β_0, β_1, σ and τ. **WinBUGS** is executed using the **R** command `bugs()`. We set the option `debug=TRUE` if we want the **WinBUGS** GUI to stay open (useful for analyzing MCMC output and looking at the **WinBUGS** error log). Also, we set `working.dir=getwd()` so that **WinBUGS** output files and the log file are saved in the current **R** working directory (note that sometimes you will need to specify the place where you installed **WinBUGS** within the `bugs()` call, using the `bugs.directory` argument). All of these activities together look like this:

```
> library(R2WinBUGS)    # "load" the R2WinBUGS package
> data <- list( y=y, x=x)
> inits <- function()
> list ( beta1=rnorm(1),beta0=rnorm(1),sigma=runif(1,0,2) )
> parameters <- c("beta0","beta1","sigma","tau")
> out <- bugs(data, inits, parameters, "normal.txt", n.thin=1, n.chains=2,
            n.burnin=2000, n.iter=6000, debug=TRUE,working.dir=getwd())
```

Note that the previously created objects defining data, initial values and parameters to monitor are passed to the function `bugs()`. In addition, various other things are declared: The number of parallel Markov chains (`n.chains`), the thinning rate (`n.thin`), the number of burn-in iterations (`n.burnin`) and the total number of iterations (`n.iter`). To develop a detailed understanding of the various parameters and settings used for MCMC, consult a basic reference such as Kéry (2010). We also come back to these issues in the following section (3.5) and in Chapt. 17. A common question is "how should my data be formatted?" That depends on how you describe the model in the **BUGS** language, and how your data are input into **R**. There is no unique way to describe any particular model and so you have some flexibility. We talk about data format further in the context of capture-recapture models and SCR models in Chapt. 5 and elsewhere.

You should execute all of the commands given above and then close the **WinBUGS** GUI, and the data will be read back into **R** (or specify `debug=FALSE` in the `bugs()` call). We don't want to give instructions on how to navigate and use the GUI – but you can fire up **WinBUGS** and read the help files, or see Chapt. 4 from Kéry (2010) for a brief introduction. The print command applied to the object `out` prints some basic summary output (this is slightly edited):

```
> print(out,digits=2)
Inference for Bugs model at "normal.txt", fit using WinBUGS,
 2 chains, each with 6000 iterations (first 2000 discarded)
```

[5]While **WinBUGS** is reasonably robust to a wide range of more or less plausible starting values, **JAGS** is a lot more sensitive and especially with more complex models you might actually have to spend some time thinking about how to specify good starting values to get the model running (Appendix 1); we will come back to this issue when we use **JAGS**

```
 n.sims = 8000 iterations saved
           mean   sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
beta0    -6.62 1.64 -9.77 -7.63 -6.64 -5.63 -3.29    1  4200
beta1     0.81 1.20 -1.63  0.09  0.80  1.54  3.24    1  5100
sigma     4.99 1.56  2.93  3.92  4.66  5.70  8.85    1  8000
tau       0.05 0.03  0.01  0.03  0.05  0.07  0.12    1  8000
deviance 58.72 3.21 55.06 56.35 57.85 60.26 67.15    1  6200

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = Dbar-Dhat)
pD = 2.5 and DIC = 61.3
```

In the **WinBUGS** output you see a column called "`Rhat`", as well as one called "`n.eff`". These are convergence diagnostics (the $\hat{R}$ or Brooks-Gelman-Rubin statistic and the effective sample size) and we will discuss those in the following section, 3.5.2. DIC is the deviance information criterion (Spiegelhalter et al. (2002), see section 3.9) which some people use in a manner similar to AIC although it is recognized to have some problems in hierarchical models (Millar, 2009). We consider use of DIC in the context of SCR models in Chapt. 8.

3.5 PRACTICAL BAYESIAN ANALYSIS AND MCMC

The mere execution of a Bayesian analysis using the **BUGS** language, as demonstrated with the linear regression example, is fairly straight forward. There are, however, a number of really important practical issues to be considered in any Bayesian analysis and we cover some of these briefly here before we move on to implementing slightly more complex GL(M)Ms in a Bayesian framework.

3.5.1 Choice of prior distributions

Bayesian analysis requires that we choose prior distributions for all of the structural parameters of the model (we use the term structural parameter to mean all parameters that aren't customary thought of as latent variables). We will strive to use priors that are meant to express little or no prior information - default or customary "non-informative" or diffuse priors. This will be Uniform(a, b) priors for parameters that have a natural bounded support and, for parameters that live on the real line we use either (1) diffuse normal priors, as we did in the linear regression example above; (2) improper uniform priors which have unbounded support, e.g., $[\theta] \propto 1$, or (3) sometimes even a bounded Uniform(a, b) prior, if that greatly improves the performance of **WinBUGS** or other software doing the MCMC for us. In **WinBUGS** a prior with low precision, τ, where $\tau = 1/\sigma^2$, such as Normal$(0, .01)$ will typically be used. Of course $\tau = 0.01$ ($\sigma^2 = 100$) might be very informative for a regression parameter depending on its magnitude and scaling of x. Therefore, we recommend that predictor variables (covariates) *always* be standardized to have mean 0 and variance 1.

Lack of invariance of priors to transformation. Clearly there are a lot of choices for ostensibly non-informative priors, and the degree of non-informativeness depends on the parameterization. For example, a natural non-informative prior for the intercept of a logistic regression

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

would be a very diffuse normal prior, $[\beta_0] = \text{Normal}(0, \text{Large})$ or even $\beta_0 \sim \text{Uniform}(-\text{Large}, \text{Large})$. However, we might also use a prior on the parameter $p_0 = \text{logit}^{-1}(\beta_0)$, which is $\Pr(y = 1)$ for the value $x = 0$. Since p_0 is a probability a natural choice is $p_0 \sim \text{Uniform}(0, 1)$. These priors are very different in their implications. For example, if we choose the normal prior for β_0 with variance Large $= 5^2$ and look at the implied prior for p_0 we have the result shown in Fig. 3.2 which looks nothing like a Uniform$(0, 1)$ prior. These two priors can

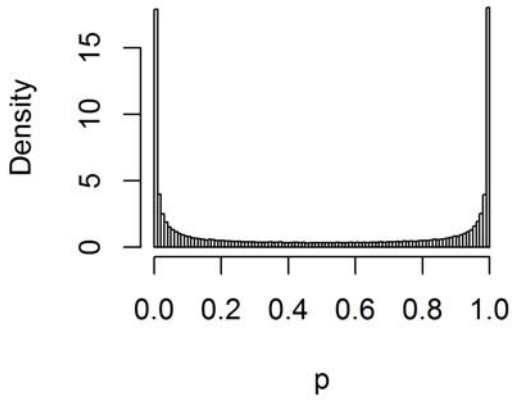

Figure 3.2. Implied prior for $p_0 = \exp(\beta_0)/(1 + \exp(\beta_0))$ if $\beta_0 \sim \text{Normal}(0, 5^2)$.

affect results (see Sec. 4.4.2 for an illustration of this for a real data set), yet they are both sensible non-informative priors. Despite this, it is often the case that priors will have little or no impact on the results. Choice of priors and parameterization is very much problem-specific and often largely subjective. Moreover, it also affects the behavior of MCMC algorithms and therefore the analyst needs to pay some attention to this issue and possibly try different things out. Most standard Bayesian analysis books address issues related to specification and effect of prior distribution choice in some depth. Some good references include Kass and Wasserman (1996), Gelman (2006) and Link and Barker (2010).

3.5.2 Convergence and so-forth

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, the generated samples can be used to characterize the posterior distribution. Thus, one of the most important issues 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 (this is the whole point of doing an MCMC analysis), 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. Also, you only know the part of your posterior distribution that the Markov chain has explored so far – for all you know the chain could be stuck in a local maximum, while other maxima remain completely undiscovered. Acknowledging that there is truly nothing we can do to ever prove convergence of our MCMC chains, there are several things we can do to increase the degree of confidence we have about the convergence of our 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. Fig. 3.3 shows the time series plots for the three parameters – β_0, β_1 and σ – from our linear regression example, taken from the **WinBUGS** GUI before closing it to return to **R**.

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. In our linear regression example, within the `bugs()` call we set the burn-in period as 2000 iterations so these are automatically removed by **WinBUGS** and are not part of the output (but Fig. 3.6 shows a time-series plot that starts at iteration 0 with a clearly visible burn-in period). The quick diagnostic to whether convergence has been achieved is that your Markov chains look "grassy" – this seems a reasonable statement for the plots in Fig. 3.3. 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, and one generally implemented in **WinBUGS**, is to run several Markov chains and to start them off at different initial values that are over-dispersed relative to the posterior distribution. Such initial values 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 came up with the so-called "R-hat" statistic ($\hat{R}$) or Brooks-Gelman-Rubin statistic that essentially compares within-chain and between-chain variance to check for convergence of multiple chains (Gelman et al., 1996). The R-hat statistic should be close to 1 if the Markov chains have converged and sufficient posterior samples have been obtained. For the linear regression example, we ran two parallel chains (also specified in the `bugs()` call) and **WinBUGS** returns the $\hat{R}$ statistic for us as part of the summary model output. If you look back to Sec. 3.4.1 you see that $\hat{R} = 1$ for all parameters of the linear model. In practice, $\hat{R} \leq 1.2$ may be good enough for some problems. For some models you can't actually realize a low $\hat{R}$. E.g., if the posterior is a discrete mixture of distributions

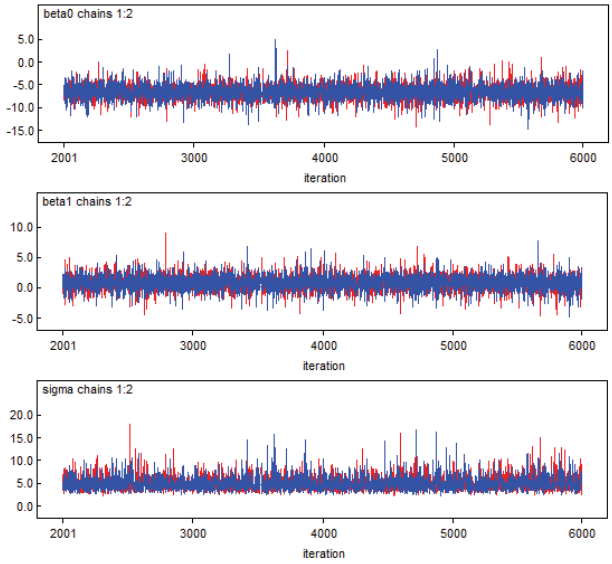

Figure 3.3. Time-series plots for parameters from a linear regression run in **WinBUGS** using two parallel Markov chains.

then you can be misled into thinking that your Markov chains have not converged when in fact the chains are just jumping back and forth in the posterior state-space. This happens in some of indicator variable model selection discussed in Chapt. 8. Often, when there is little information about a parameter in the data, or when parameters are on the boundary of the parameter space, convergence will appear to be poor also. These kinds of situations are normally ok and you need to think really hard about the context of the model and the problem before you conclude that your MCMC algorithm is ill-behaved.

Some models exhibit "poor mixing" of the Markov chains (or "slow convergence") 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 rather slowly. Poor mixing can happen for many reasons – when parameters are highly correlated (even confounded), or barely identified from the data, or the algorithms are very terrible and probably 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 kept. 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, 2011). Practical considerations might necessitate thinning, even though it is statistically inefficient. For example, in models with many parameters or other unknowns being tabulated, 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 the investigator should experiment with different settings and remain calm when things don't work out perfectly.

Is the posterior sample large enough? 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[6] and the sample size has to be adjusted to account for the autocorrelation in subsequent samples (see Chapt. 8 in Robert and Casella (2010) for more details). 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. **WinBUGS** will automatically return the effective sample size for all monitored parameters, as we saw in our linear regression example (the "n.eff" column of the summary output). If you find that your supposedly long Markov chain has only generated a very short effective sample, you should consider a longer run. What exactly constitutes a reasonable effective sample size is hard to say. A more palpable measure of whether you've run your chain for enough iterations is the time-series or Monte Carlo 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, which can be reproduced in **R** using `bugs.log('log.txt')$stats` (note that "log.txt" refers to a model log file that **WinBUGS** automatically creates in the working directory; it is overwritten with every new model you run unless you save it under a different name).

```
> bugs.log('log.txt')$stats
$stats
              mean      sd   MCerror    2.5%   median   97.5% start sample
beta0    -6.64700 1.60300 0.0179400 -9.7140 -6.70800 -3.2730  2001   8000
beta1     0.82100 1.19000 0.0116800 -1.4900  0.82560  3.1800  2001   8000
deviance 58.66000 3.08800 0.0506800 55.0700 57.93000 66.8400  2001   8000
sigma     4.96800 1.52300 0.0248300  2.9350  4.68100  8.7410  2001   8000
tau       0.05074 0.02677 0.0003651  0.0131  0.04564  0.1162  2001   8000
```

When using **JAGS** the `summary` command will automatically produce the MC error (which is called "Time-series SE" in **JAGS**). 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. 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 2 or 3 significant digits!

[6] In case you are not familiar with Markov chains, for T random samples $\theta^{(1)}$, ... $\theta^{(T)}$ from a Markov chain the distribution of $\theta^{(t)}$ depends only on the immediately preceding value, $\theta^{(t-1)}$.

3.5.3 Bayesian confidence intervals

The 95% Bayesian confidence interval based on percentiles of the posterior is not a unique interval - there are many of them. The so-called "highest posterior density" (HPD) interval is an alternative, defined as the narrowest interval that contains *at least* 95% of the posterior mass. As a result (of the *at least* clause), for discrete parameters, the 95% HPD is not often exactly 95% but usually slightly more conservative than nominal.

3.5.4 Estimating functions of parameters

A benefit of analysis by MCMC is that we can seamlessly estimate functions of parameters by simply tabulating the desired function of the simulated posterior draws. For example, if θ is the parameter of interest and let $\theta^{(i)}$ for $i = 1, 2, \ldots, M$ be the posterior samples of θ. Let $\eta = \exp(\theta)$, then a posterior sample of η can be obtained simply by computing $\exp(\theta^{(i)})$ for $i = 1, 2, \ldots, M$. Almost all SCR models in this book involve at least 1 derived parameter. For example, density D is a derived parameter, being a function of population size N and the area A of the underlying state-space of the point process (see Chapt. 5).

Example: Finding the optimum value of a covariate. As another example of estimating functions of model parameters, suppose that the normal regression model from Sec. 3.4.1 had a quadratic response function of the form

$$\mathbb{E}(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2.$$

Then the optimum value of x, i.e., that corresponding to the optimal expected response, can be found by setting the derivative of this function to 0 and solving for x. We find that

$$df/dx = \beta_1 + 2 * \beta_2 x = 0$$

yields that $x_{opt} = -\beta_1/(2 * \beta_2)$. We can just take our posterior draws for β_1 and β_2 and obtain a posterior sample of x_{opt} by this simple calculation applied to the posterior output. As an exercise, take the normal model above and simulate a quadratic response and then describe the posterior distribution of x_{opt}.

3.6 POISSON GLMS

The Poisson GLM (also known as "Poisson regression") is probably the most relevant and important class of models in all of ecology. The basic model assumes observations $y_i; i = 1, 2, ..., n$ follow a Poisson distribution with mean λ which we write

$$y_i \sim \text{Poisson}(\lambda)$$

Commonly y_i is a count of animals or plants at some point in space ("site") i, and λ might vary over sites as well. For example, i might index point count locations in a forest, survey route centers, or sample quadrats, or similar, and we are interested in how λ depends on site characteristics such as habitat. If covariates are available it is typical to model them as linear effects on the log mean. If x_i is some measured covariate associated with observation i, then,

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

While we only specify the mean of the Poisson model directly, the Poisson model (and all GLMs) has a "built-in" variance which is directly related to the mean. In this case, $\mathrm{Var}(y) = \mathbb{E}(y) = \lambda$. Thus the model accommodates a linear increase in variance with the mean.

3.6.1 Example: Breeding Bird Survey data

As an example we consider a classical situation in ecology where counts of an organism are made at a collection of spatial locations. In this particular example, we have mourning dove (*Zenaida macroura*) counts made along North American Breeding Bird Survey (BBS) routes in Pennsylvania, USA. A route consists of 50 stops separated by 0.5 miles. For the purposes here we are defining y_i = route total count and the sample location will be marked by the center point of the BBS route. The survey is run annually and the data set we analyze is 1966-1998. BBS data can be obtained online at `http://www.pwrc.usgs.gov/bbs/`, but the particular chunk of data we will be using here is also included in the `scrbook` package (`data(bbsdata)`). We will make use of the whole data set shortly but for now we're going to focus on a specific year of counts (1990) for the sake of building a simple model. In 1990 there were 77 active routes; this data set contains rows which index the unique route, column 1 is the route ID, columns 2-3 are the route coordinates (longitude/latitude), column 4 is a habitat covariate "forest cover" (standardized, see below) and the remaining columns are the yearly counts. Years for which a survey was not conducted on a route are coded as "`NA`" in the data matrix. We imagine that this will be a typical format for many ecological studies, perhaps with more columns representing covariates. To read in the data and display the first few elements of the data frame containing the counts, do this:

```
> data(bbsdata)               #  loads data frame 'bbs'
> bbsdata$counts[1:2,1:6]

      X     lon    lat    habitat X66 X67
1 72002 -80.445 41.501 -0.3871372  NA  24
2 72003 -80.347 41.214 -1.0171629  NA  NA
```

It is useful to display the spatial pattern in the observed counts. For that we use a spatial dot plot – where we plot the coordinates of the observations and mark the color of the plotting symbol based on the magnitude of the count. We have a special plotting function for that which is called `spatial.plot()` and it is available with the supplemental **R** package `scrbook`. Actually, what we want to do here is plot the log-counts (+1 of course) which (Fig. 3.4) display a notable pattern that could be related to something. The **R** commands for obtaining this figure are:

```
> library(scrbook)
> data(bbsdata)
> library(maps)

> y <- bbsdata$counts[,"X90"]  # Pick year 1990
> notna <- !is.na(y)
```

```
> y <- y[notna]
> locs <- bbsdata$counts[notna,c("lon","lat")]
> sz <- y/max(y)

> par(mar=c(3,3,3,6))
> plot(locs,pch=" ",axes=FALSE,xlim=range(locs[,1])+c(-.3,+.3),
     ylim=c(range(locs[,2]) + c(-.6,.6)), xlab=" ",ylab=" ")
> map('state', regions='pennsylvania', add=TRUE, lwd=2)
> spatial.plot(bbsdata$counts[notna,2:3], y, cx=1+sz*6, add=TRUE)
```

We can ponder the potential effects that might lead to dove counts being high - corn fields, telephone wires, barn roofs along with misidentification of pigeons, these could all correlated reasonably well with the observed count of mourning doves. Unfortunately we don't have any of that information. However, we do have a measure of forest cover (pro-

Figure 3.4. Mourning dove counts along North American Breeding Bird Survey routes in Pennsylvania (year = 1990). Plot symbol shading and circle size is proportional to raw count.

vided in the data frame `bbsdata$habitat`) which can be plotted using the `spatial.plot` function with the following **R** commands

```
> habdata <- bbsdata$habitat
> map('state',regions="penn",lwd=2)
> I <- matrix(NA, nrow=30, ncol=40)
> I <- matrix(habdata[,"dfor"], ncol=40, byrow=FALSE)
> ux <- unique(habdata[,2])
```

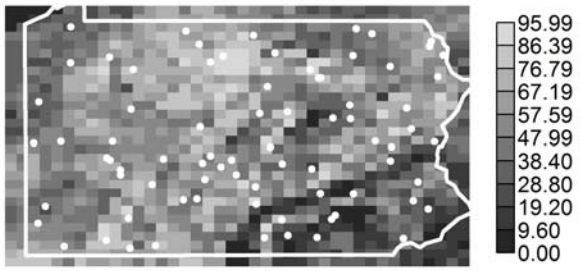

Figure 3.5. Forest cover (percent deciduous) in Pennsylvania. BBS route locations are shown by white dots.

```
> uy <- sort(unique(habdata[,3]))

> par(mar=c(3,3,3,6))
> plot(locs,pch=" ", axes=FALSE, xlim=range(locs[,1])+c(-.3,+.3),
     ylim=c(range(locs[,2]) + c(-.6,.6)), xlab=" ",ylab=" ")
> image(ux,uy,rot(I), add=TRUE, col=gray(seq(3,17,,10)/20) )
> map('state', regions='pennsylvania', add=TRUE, lwd=3, col="white")
> image.scale(I, col=gray(seq(3,17,,10)/20) )
> points(locs,pch=20, col="white")
```

The result appears in Fig. 3.5. We see a prominent pattern that indicates high forest coverage in the central part of the state and low forest cover in the SE. Inspecting the previous figure of the raw counts suggests a relationship between counts and forest cover which is perhaps not surprising.

3.6.2 Doing it in WinBUGS

Here we demonstrate how to fit a Poisson GLM in **WinBUGS** using the covariate $x_i =$ forest cover along BBS route i. It is advisable that x_i be standardized in most cases as this will improve mixing of the Markov chains. We have pre-standardized the forest cover covariate for the BBS route locations, and so we don't have to worry about that here. To read the BBS data into **R** and get things set up for **WinBUGS** we issue the following commands:

```
> library(scrbook)
> data(bbsdata)

> y <- bbsdata$counts[,"X90"]               # Pick year 1990
> notna <- !is.na(y)
> y <- y[notna]

## Forest cover already standardized here:
> habitat <- bbsdata$counts[notna,"habitat"]
> M <- length(y)

> library(R2WinBUGS)                        # Load R2WinBUGS
> data <- list (y=y, M=M, habitat=habitat)  # Bundle data for WinBUGS
```

Now we write out the Poisson model specification in **WinBUGS** pseudo-code, provide initial values, identify parameters to be monitored and then execute **WinBUGS**:

```
> cat("
model{
    for (i in 1:M){
      y[i] ~ dpois(lam[i])
      log(lam[i]) <- beta0+beta1*habitat[i]
     }
 beta0 ~ dunif(-5,5)
 beta1 ~ dunif(-5,5)
}
",file="PoissonGLM.txt")

> inits <- function()  list ( beta0=rnorm(1),beta1=rnorm(1) )
> parameters <- c("beta0","beta1")
> out <- bugs(data, inits, parameters, "PoissonGLM.txt", n.thin=2,n.chains=2,
          n.burnin=2000,n.iter=6000,debug=TRUE,working.dir=getwd())
```

The **WinBUGS** output can be viewed in **R** using the `print` command:

```
print(out,digits=2)
Inference for Bugs model at "PoissonGLM.txt", fit using WinBUGS,
 2 chains, each with 6000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 4000 iterations saved
             mean   sd    2.5%     25%     50%     75%   97.5% Rhat n.eff
beta0       3.15 0.02    3.10    3.13    3.15    3.17    3.20    1  4000
beta1      -0.50 0.02   -0.54   -0.51   -0.50   -0.48   -0.46    1  4000
deviance 1116.56 1.95 1115.00 1115.00 1116.00 1117.00 1122.00    1  4000
```

3.6.3 Constructing your own MCMC algorithm

At this point it might be helpful to suffer through an example building a custom MCMC algorithm. Here, we develop an MCMC algorithm for the Poisson regression model, using

a Metropolis-within-Gibbs sampling framework. Building MCMC algorithms is covered in more detail in Chapt. 17 where you can also find step-by-step instructions for Metropolis-within-Gibbs samplers, should the following section move through all this material too quickly.

We will assume that the two parameters, β_0 and β_1,have diffuse normal priors, say $[\beta_0] = \text{Normal}(0, 100)$ and $[\beta_1] = \text{Normal}(0, 100)$ where each has *standard deviation* 100 (recall that **WinBUGS** parameterizes the normal in terms of $1/\sigma^2$). We need to assemble the relevant elements of the model which are these two prior distributions and the likelihood $[\mathbf{y}|\beta_0, \beta_1] = \prod_i [y_i|\beta_0\beta_1]$ which is, mathematically, the product of the Poisson pmf evaluated at each y_i, given particular values of β_0 and β_1. Next, we need to identify the full conditionals $[\beta_0|\beta_1, \mathbf{y}]$ and $[\beta_1|\beta_0, \mathbf{y}]$. We use the all-purpose rule for constructing full conditionals (section 3.3.2) to discover that:

$$[\beta_0|\beta_1, \mathbf{y}] \propto \left\{ \prod_i [y_i|\beta_0, \beta_1] \right\} [\beta_0]$$

Mathematically, the full conditional is of the form

$$[\beta_0|\beta_1, \mathbf{y}] \propto \left\{ \prod_i \exp(-\exp(\beta_0 + \beta_1 x_i)) \exp(\beta_0 + \beta_1 x_i)^{y_i} \right\} \exp(-\frac{\beta_0^2}{2 * 100})$$

which you can program as an **R** function with arguments β_0, β_1 and $\mathbf{y}$ without difficulty. The full-conditional for β_1 is:

$$[\beta_1|\beta_0, \mathbf{y}] \propto \left\{ \prod_i [y_i|\beta_0, \beta_1] \right\} [\beta_1]$$

which has a similar mathematical representation except the prior is expressed in terms of β_1 instead of β_0. Remember, we could replace the "$\propto$" with "$=$" if we put $[y|\beta_1]$ or $[y|\beta_0]$ in the denominator. But, in general, $[y|\beta_0]$ or $[y|\beta_1]$ will be quite a pain to compute and, more importantly, it is a constant as far as the operative parameters (β_0 or β_1, respectively) are concerned. Therefore, the MH acceptance probability will be the ratio of the full-conditional evaluated at a candidate draw to that evaluated at the current value, and so the denominator required to change $\propto$ to $=$ winds up canceling from the MH acceptance probability.

Here we will use the so-called random walk candidate generator, which is a Normal proposal distribution, so that, for example, $\beta_0^* \sim \text{Normal}(\beta_0^t, \delta)$ where δ is the standard-deviation of the proposal distribution, which is just a tuning parameter that is set by the user and adjusted to achieve efficient mixing of chains (see Sec. 17.2.2). We remark also that calculations are often done on the log-scale to preserve numerical integrity of things when quantities evaluate to small or large numbers, so keep in mind, for example, $a * b = \exp(\log(a) + \log(b))$ for two positive numbers a and b. The "Metropolis within Gibbs" algorithm for a Poisson regression turns out to be remarkably simple and is given in Panel 3.1. It is also part of the `scrbook` package and you can run 1000 iterations of it by calling `PoisGLMBBS(y=y, habitat=habitat, niter=1000)` (note that `y` = point count data and `habitat` = forest cover have to be defined in your **R** workspace as shown in the previous analysis of these data).

```
> set.seed(2013)      # So we all get the same result

> out <- matrix(NA,nrow=1000,ncol=2)   # Matrix to store the output
> beta0 <- -1                          # Starting values
> beta1 <- -.8

# Begin the MCMC loop ; do 1000 iterations
> for(i in 1:1000){

# Update the beta0 parameter
lambda <- exp(beta0+beta1*habitat)
lik.curr <- sum(log(dpois(y,lambda)))
prior.curr <- log(dnorm(beta0,0,100))
beta0.cand <- rnorm(1,beta0,.05)         # generate candidate
lambda.cand <- exp(beta0.cand + beta1*habitat)
lik.cand <- sum(log(dpois(y,lambda.cand)))
prior.cand <- log(dnorm(beta0.cand,0,100))
mhratio <- exp(lik.cand +prior.cand - lik.curr-prior.curr)
if(runif(1)< mhratio)
     beta0 <- beta0.cand

# update the beta1 parameter
lik.curr <- sum(log(dpois(y,exp(beta0+beta1*habitat))))
prior.curr <- log(dnorm(beta1,0,100))
beta1.cand <-rnorm(1,beta1,.25)
lambda.cand <- exp(beta0+beta1.cand*habitat)
lik.cand <- sum(log(dpois(y,lambda.cand)))
prior.cand <- log(dnorm(beta1.cand,0,100))
mhratio <- exp(lik.cand + prior.cand - lik.curr - prior.curr)
if(runif(1)< mhratio)
     beta1 <- beta1.cand

out[i,] <- c(beta0,beta1)              # save the current values
}

> plot(out[,1],ylim=c(-1.5,3.3),type="l",lwd=2,ylab="parameter value",
     xlab="MCMC iteration")
> lines(out[,2],lwd=2,col="red")
```

Panel 3.1: **R** code to run a Metropolis sampler on a simple Poisson regression model.

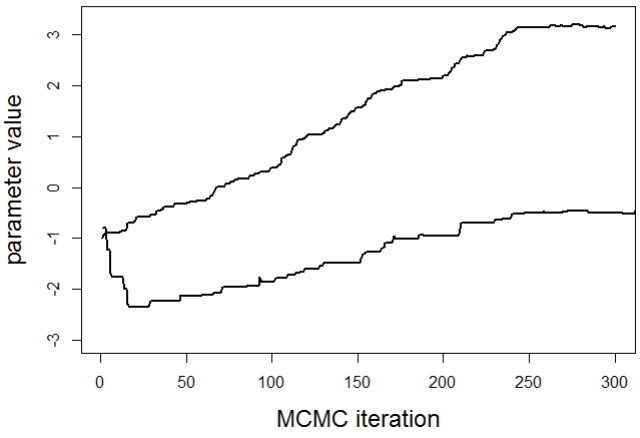

Figure 3.6. First 300 MCMC iterations for the Poisson GLM model parameters β_0 (top) and β_1 (bottom) using a Metropolis-Hastings tuning parameter of $\delta = 0.05$.

The first 300 iterations of the MCMC history of each parameter are shown in Fig. 3.6. These chains are not very appealing but a couple of things are evident: We see that the burn-in takes about 250 iterations and that after that chains seem to mix reasonably well, although this is not so clear given the scale of the y-axis, which we have chosen to get both variables on the same graph. We generated 10,000 posterior samples, discarding the first 500 as burn-in, and the result is shown in Fig. 3.7, this time on separate panels for each parameter. The "grassy" look of the MCMC history is diagnostic of Markov chains that are well-mixing and we would generally be very satisfied with results that look like this.

Note that we used a specific set of starting values for these simulations. It should be clear that starting values closer to the mass of the posterior distribution might cause burn-in to occur faster. Note also that we have used a different prior than in our **WinBUGS** model specification given previously. We encourage you to evaluate whether this seems to affect the result.

3.7 POISSON GLM WITH RANDOM EFFECTS

In most of this book, we will be dealing with random effects in GLM-like models – similar to what are usually referred to as generalized linear mixed models (GLMMs). We provide a brief introduction of such a model by way of example, extending our Poisson regression model to include a random effect.

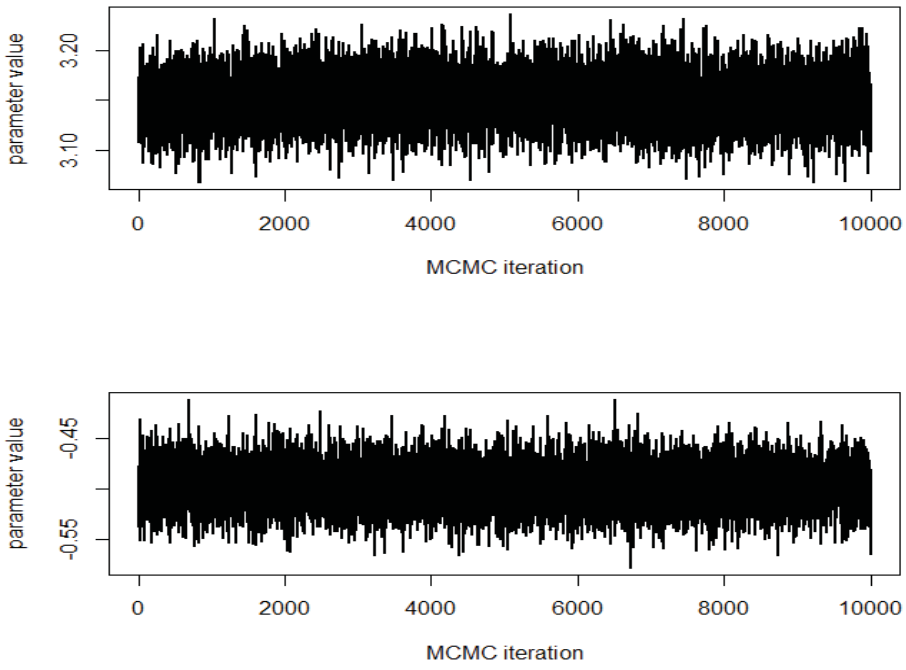

Figure 3.7. Nice grassy plots of 10,000 MCMC iterations for the Poisson GLM model parameters β_0 (top) and β_1 (bottom) using a Metropolis-Hastings tuning parameter of $\delta = 0.05$.

The Log-Normal mixture: The classical situation involves a GLM with a normally distributed random effect that is additive on the linear predictor. For the Poisson case, we have:

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

where $\eta_i \sim \text{Normal}(0, \sigma^2)$. In this context, η could represent an error term capturing variation in λ_i not accounted for by the covariates, or overdispersion. It is really amazingly simple to express this model in the **BUGS** language and have **WinBUGS** (or **JAGS**, etc..) draw samples from the posterior distribution. The code for analysis of the BBS dove counts is given as follows:

```
> library(scrbook)
### Grab the BBS Data as before
> data(bbsdata)
### Set random seed so that results are repeatable
```

Table 3.1. Posterior summaries for Poisson GLMM containing a normal random effect and a habitat effect for mourning dove counts across BBS routes in PA, 1990. Model was fit using **WinBUGS**, 2 chains, each with 5000 iterations (first 1000 discarded), n.thin = 2 n.sims = 4000 iterations saved.

Parameter	Mean	SD	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
β_0	2.98	0.08	2.82	2.93	2.98	3.03	3.12	1.00	1400
β_1	-0.53	0.07	-0.68	-0.58	-0.53	-0.49	-0.38	1.01	350
σ	0.60	0.06	0.49	0.56	0.59	0.64	0.73	1.00	2000
τ	2.88	0.57	1.88	2.47	2.86	3.24	4.12	1.00	2000
deviance	445.94	12.18	424.00	437.40	445.20	453.90	471.50	1.00	4000

```
> set.seed(2013)
### Dump the BUGS model into a file
> cat("
model{
  for (i in 1:M){   # Observation model, linear predictor, etc..
     y[i] ~ dpois(lam[i])
     log(lam[i]) <- beta0+ beta1*habitat[i] + eta[i]
     frog[i] <- beta1*habitat[i] + eta[i]
     eta[i] ~ dnorm(0,tau)
   }
                    # Prior distributions:
 beta0 ~ dunif(-5,5)
 beta1 ~ dunif(-5,5)
 sigma ~ dunif(0,10)
 tau <- 1/(sigma*sigma)
}
",file="model.txt")
```

```
> data <- list ("y","M","habitat")  # Define the data
> inits <- function()               #    inits and parameters
  list ( beta0=rnorm(1), beta1=rnorm(1), sigma=runif(1,0,4))
> parameters <- c("beta0","beta1","sigma","tau")

> library(R2WinBUGS)                # Load and run R2WinBUGS
> out <- bugs (data, inits, parameters, "model.txt", n.thin=2,n.chains=2,
               n.burnin=1000, n.iter=5000, debug=TRUE)
```

This produces the posterior summary statistics given in Table 3.1. One thing we notice is that the posterior standard deviations of the regression parameters are much higher, a result of the extra-Poisson variation allowed for by this model. We would also notice much less precise predictions of hypothetical new observations.

3.8 BINOMIAL GLMS

Another extremely important class of models in ecology are binomial models. We use binomial models for count data whenever the observations are counts or frequencies and it is natural to condition on a "sample size", say K, the maximum frequency possible in a sample. The random variable, $y \leq K$, is then the frequency of occurrences out of K "trials". The parameter of the binomial models is p, often called "success probability" which is related to the expected value of y by $\mathbb{E}(y) = pK$. Usually we are interested in modeling covariates that affect the parameter p, and such models are called binomial GLMs, binomial regression models or logistic regression, although logistic regression really only applies when the logistic link is used to model the relationship between p and covariates (see below).

One of the most typical binomial GLMs occurs when the sample size equals 1 and the outcome, y, is "presence" ($y = 1$) or "absence" ($y = 0$) of a species. In this case, y has a Bernoulli distribution. This is a classical species distribution modeling situation. A special situation occurs when presence/absence is observed with error (MacKenzie et al., 2002; Tyre et al., 2003). In that case, $K > 1$ samples are usually needed for effective estimation of model parameters.

In standard binomial regression problems the sample size is fixed by design but interesting models also arise when the sample size is itself a random variable. These are the N-mixture models (Royle, 2004b; Kéry et al., 2005; Royle and Dorazio, 2008; Kéry, 2010) and related models (in this case, N being the sample size, which we labeled K above)[7]. Another situation in which the binomial sample size is "fixed" is closed population capture-recapture models in which a population of individuals is sampled K times. The number of times each individual is encountered is a binomial outcome with parameter (encounter probability) p, based on a sample of size K. In addition, the total number of unique individuals observed, n, is also a binomial random variable based on population size N. We consider such models in Chapt. 4.

3.8.1 Binomial regression

In binomial models, covariates are modeled on a suitable transformation (the link function) of the binomial success probability, p. Let x_i denote some measured covariate for sample unit i and let p_i be the success probability for unit or subject i. The standard choice is the logit link function (3.1) but there are many other possible link functions. We sometimes use the complementary log-log (= "cloglog") link function in ecological applications because it is natural in some cases when the response should scale in relation to area or effort (Royle and Dorazio, 2008, p. 150). As an example, the "probability of observing a count greater than 0" under a Poisson model is $\Pr(y > 0) = 1 - \exp(-\lambda)$. In that case, for the i^{th} observation,

$$\text{cloglog}(p_i) = \log(-\log(1 - p_i)) = \log(\lambda_i)$$

so that if you have covariates in your linear predictor for $\mathbb{E}(y)$ under a Poisson model then they are linear on the complementary log-log link of p. In models of species occurrence

[7]Some of the jargon is actually a little bit confusing here because the binomial index is customarily referred to as "sample size" but in the context of N-mixture models N is actually the "population size"

it seems natural to view occupancy as being derived from local abundance N (Royle and Nichols, 2003; Royle and Dorazio, 2006; Dorazio, 2007). Therefore, models of local abundance in which $N_i \sim \text{Poisson}(A_i\lambda_i)$ for a habitat patch of area A_i implies a model for occupancy ψ_i of the form

$$\text{cloglog}(\psi_i) = \log(A_i) + \log(\lambda_i).$$

We will use the cloglog link in some analyses of SCR models in Chapt. 5 and elsewhere.

3.8.2 Example: waterfowl banding data

The standard binomial modeling problem in ecology is that of modeling species distributions, where $K = 1$ and the outcome is occurrence ($y = 1$) or not ($y = 0$) of some species. Such examples abound in books (e.g., Royle and Dorazio (2008, ch. 3); Kéry (2010, ch. 21); Kéry and Schaub (2012, ch. 13)) and in the literature. Therefore, instead, we will consider an example involving band returns of waterfowl in the upper great plains including some Canadian provinces, which were analyzed by Royle and Dubovsky (2001).

For these data, y_{it} is the number of mallard (*Anas platyrhynchos*) bands recovered out of B_{it} birds banded at some location $\mathbf{s}_i$ in year t. In this case B_{it} is fixed. Thinking about recovery rate as being proportional to harvest rate, we use these data to explore geographic gradients in recovery rate resulting from variability in harvest pressure experienced by different populations. As such, we fit a basic binomial GLM with a linear response to geographic coordinates (including an interaction term). Here we provide the part of the script for creating the model and fitting the model in **WinBUGS**. There are few structural differences between this model and the Poisson GLM fitted previously. The main things are due to the data structure (we have a matrix here instead of a vector) and otherwise we change the distributional assumption to binomial (specified with `dbin`) and then use the `logit` function to relate the parameter p_{it} to the covariates.

Dummy variables in BUGS: In the mallard example, we model the band recovery probability p_{it} not only as a linear function (on the logit scale) of geographic location, but also allow for variation in p_{it} with year, t; $t = 1, 2, ...T$. In this particular example there are $T = 5$ years of data and we could describe the full mallard model with a formula in terms of "dummy variables." Dummy variables are binary variables, one variable for each level of the categorical variable they describe, such that variable for level t takes on the value 1 if the observation belongs with level t and 0 otherwise. So, the mallard model in terms of dummy variables for "year" looks like this:

$$\begin{aligned} y_{it} &\sim \text{Binomial}(p_{it}, B_{it}) \\ \text{logit}(p_{it}) &= \beta_0 + \beta_1 x_{2,it} + \beta_2 x_{3,it} + \beta_3 x_{4,it} + \beta_4 x_{5,it} + \beta_5 \texttt{Lat}_i + \beta_6 \texttt{Lon}_i + \beta_7 \texttt{Lat}_i\texttt{Lon}_i \end{aligned}$$

Here, x_2 to x_5 are the dummy variable vectors of length T that take on the value of 1 when t corresponds to the respective year and 0 otherwise; β_0 is the common intercept term and corresponds to $t = 1$; β_1 - β_4 describe the difference in p_{it} for each t relative to $t = 1$.

There is a more concise way of implementing such a model with a categorical covariate in **BUGS**, namely, by using indexing instead of dummy variables[8]. Essentially, instead of estimating the difference in p relative to category 1, we estimate a separate intercept term for each category, so that we have 5 different β_0 parameters indexed by t. This reduces the linear predictor to:

$$\text{logit}(p_{it}) = \beta_{0t} + \beta_5 \texttt{Lat}_i + \beta_6 \texttt{Lon}_i + \beta_7 \texttt{Lat}_i \texttt{Lon}_i$$

The model can be implemented in the **BUGS** language for the mallard banding data using the following **R** script, provided in the `scrbook` package (see `help(mallard)`):

```
> library(scrbook)
> data(mallard)    # Load mallard data

> cat("
model{
 for(t in 1:5){
    for (i in 1:nobs){
       y[i,t] ~ dbin(p[i,t], B[i,t])
       pl[i,t] <- beta0[t]+beta1*X[i,1]+beta2*X[i,2]+beta3*X[i,1]*X[i,2]
       p[i,t] <- exp(pl[i,t])/(1+exp(pl[i,t]))
     }
 }
beta1 ~ dnorm(0,.001)
beta2 ~ dnorm(0,.001)
beta3 ~ dnorm(0,.001)
for(t in 1:5){
  beta0[t] ~ dnorm(0,.001)
        }
}
",file="BinomialGLM.txt")

> library(R2WinBUGS)
> data <- list(B=mallard$bandings, y=mallard$recoveries,
              X=mallard$locs, nobs=nrow(mallard$locs))
> inits <- function(){ list(beta0=rnorm(5),beta1=0,beta2=0,beta3=0) }
> parms <- c('beta0','beta1','beta2','beta3')
> out <- bugs(data, inits, parms,"BinomialGLM.txt", n.chains=3,
              n.iter=2000, n.burnin=1000, n.thin=2, debug=TRUE)
```

Look at the posterior summaries of model parameters in Table 3.2. The basic result suggests a negative east-west gradient and a positive south to north gradient of band recovery probabilities, but no interaction. A map of the response surface is shown in Fig. 3.8.

[8] Actually, in some cases a model may mix or converge better depending on whether you choose a dummy variable or an indexing description of it, although they are structurally equivalent (Kéry, 2010)

Table 3.2. Posterior summaries for the binomial GLM of mallard band recovery rate. Model contains year-specific intercepts (β_{0t}) and a linear response surface with interaction. Model was fit using **WinBUGS**, and posterior summaries are based on 3 chains, each with 2000 iterations (first 1000 discarded), n.thin = 2 n.sims = 1500 iterations saved.

Parameter	Mean	SD	2.5%	50%	97.5%	Rhat	n.eff
$\beta_0[1]$	-2.346	0.036	-2.417	-2.346	-2.277	1.001	1500
$\beta_0[2]$	-2.356	0.032	-2.420	-2.356	-2.292	1.001	1500
$\beta_0[3]$	-2.220	0.035	-2.291	-2.219	-2.153	1.001	1500
$\beta_0[4]$	-2.144	0.039	-2.225	-2.143	-2.068	1.000	1500
$\beta_0[5]$	-1.925	0.034	-1.990	-1.924	-1.856	1.004	570
β_1	-0.023	0.003	-0.028	-0.023	-0.018	1.001	1500
β_2	0.020	0.006	0.009	0.020	0.031	1.001	1500
β_3	0.000	0.001	-0.002	0.000	0.002	1.001	1500
deviance	1716.001	4.091	1710.000	1715.000	1726.000	1.001	1500

3.9 BAYESIAN MODEL CHECKING AND SELECTION

In general terms, model checking – or assessing the adequacy of the model – and model selection are quite thorny issues and, despite contrary and, sometimes, strongly held belief among practitioners, there are not really definitive, general solutions to either problem. We're against dogma on these issues and think people need to be open-minded about such things and recognize that models can be useful whether or not they pass certain statistical tests. Some models are intrinsically better than others because they make more biological sense or foster understanding or achieve some objective that some bootstrap or other goodness-of-fit test can't decide for you. That said, it gives you some confidence if your model seems adequate in a purely statistical sense. We provide a very brief overview of concepts here, but provide more detailed coverage in Chapt. 8. See also coverage of these topics in Kéry (2010) and Link and Barker (2010) for specific context related to Bayesian model checking and selection.

3.9.1 Goodness-of-fit

Goodness-of-fit testing is an important element of any analysis because our model represents a general set of hypotheses about the ecological and observation processes that generated our data. Thus, if our model "fits" in some statistical or scientific sense, then we believe it to be consistent with the hypotheses that went into the model. More formally, we would conclude that the data are *not inconsistent* with the hypotheses, or that the model appears adequate. If we have enough data, then of course we will reject any set of statistical hypotheses. Conversely, we can always come up with a model that fits by making the model extremely complex. Despite this paradox, it seems to us that simple models that you can understand should usually be preferred even if they don't fit, for example if they embody essential mechanisms central to our understanding of things, or if we think that some contributing factors to lack-of-fit are minor or irrelevant to the scientific context and intended use of the model. In other words, models can be useful

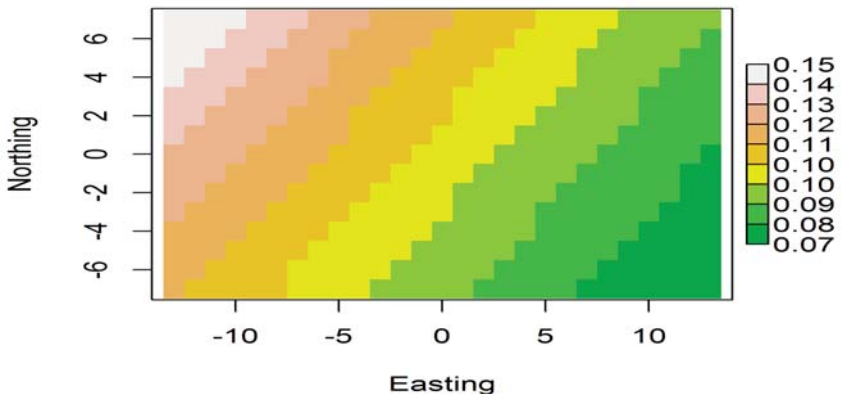

Figure 3.8. Predicted recovery rates of mallard bands in the upper great plains of North America. Note the negative gradient from the NW to the SE.

irrespective of whether they fit according to some formal statistical test of fit. Yet the tension is there to obtain fitting models, and this comes naturally at the expense of models that can be easily interpreted and studied and effectively used. Unfortunately, conducting a goodness-of-fit test is not always so easy to do. And, moreover, it is never really easy (or especially convenient) to decide if your goodness-of-fit test is worth anything. It might have 0 power! Despite this, we recommend attempting to assess model fit in real applications, as a general rule, and we provide some basic guidance here and some more specific to SCR models in Chapt. 8.

To evaluate goodness-of-fit in Bayesian analyses, we will most often use the Bayesian p-value (Gelman et al., 1996). The basic idea is to define a fit statistic or "discrepancy measure" and compare the posterior distribution of that statistic to the posterior predictive 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 the sum of squares of the "Pearson residuals",

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

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 same fit statistic:

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

and the Bayesian p-value is simply the posterior probability $\Pr(T(\mathbf{y}^{new}) > T(\mathbf{y}))$ which should be close to 0.50 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 0.50" as being "not too close to 0 or 1" and, as always, closeness is somewhat subjective. We're happy with anything $> .1$ and $< .9$ but might settle for $> .05$ and < 0.95. Another useful fit statistic is the Freeman-Tukey statistic, in which

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

(Brooks et al., 2000), where y_i is the observed value of observation i and $\mathbb{E}(y_i)$ its expected value. In contrast to a Chi-square discrepancy, the Freeman-Tukey statistic removes the need to pool cells with small expected values. In summary, you can see that the Bayesian p-value is easy to compute, and it is widely used as a result.

3.9.2 Model selection

In ecology, scientific hypotheses are often manifest as different models or parameters of a model, and so evaluating the importance of different models is fundamental to many ecological studies. For Bayesian model selection we typically use three different methods: First is, let's say, common sense. If a variable should plausibly be relevant to explaining the data-generating processes, and it has posterior mass concentrated away from 0, then it seems like it should be regarded as important - that is, it is "significant." This approach seems to have fallen out of favor in ecology over the last 10 or 15 years but in many situations it is a reasonable thing to do.

For regression problems we sometimes use the indicator variable method of Kuo and Mallick (1998), in which we introduce a set of binary variables w_k for variable k, and express the model as, e.g., for a single covariate model:

$$\mathbb{E}(y_i) = \beta_0 + w_1 \beta_1 x_i$$

where w_1 is given a Bernoulli prior distribution with some prescribed probability. E.g., $w_1 \sim$ Bernoulli(0.50) to provide a prior probability of 0.50 that variable x should be an element of the linear predictor. The posterior probability of the event $w_1 = 1$ is a gage 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. Expansion of the model to include the binary variable w_1 defines a set of 2 distinct models for which we can directly compute the posterior probabilities for, merely by tallying up the posterior frequency of w_1. See Royle and Dorazio (2008, Chapt. 3) for an example in the context of logistic regression.

This approach seems to even work sometimes with fairly complex hierarchical models of a certain form. E.g., Royle (2008) applied it to a random effects model to evaluate the importance of the random effect component of the model. The main problem, which is

2994 really a general problem in Bayesian model selection, is that its effectiveness and results
2995 will typically be highly sensitive to the prior distribution on the structural parameters
2996 (e.g., see Royle and Dorazio (2008, table 3.6)). The reason for this is obvious: If $w_1 = 0$
2997 for the current iteration of the MCMC algorithm, so that β is sampled from the prior
2998 distribution, and the prior distribution is very diffuse, then extreme values of β are likely.
2999 Consequently, when the current value of β is far away from the mass of the posterior when
3000 $w_1 = 1$, then the Markov chain may only jump from $w_1 = 0$ to $w_1 = 1$ infrequently. One
3001 seemingly reasonable solution to this problem is to fit the full model to obtain posterior
3002 distributions for all parameters, and then use those as prior distributions in a "model
3003 selection" run of the MCMC algorithm (Aitkin, 1991). This seems preferable to more-or-
3004 less arbitrary restriction of the prior support to improve the performance of the MCMC
3005 algorithm.

3006 A third method that we advocate is subject-matter context. It seems that there are
3007 some situations – some models – where one should not have to do model selection because a
3008 specific model may be necessitated by the biological context of the problem, thus rendering
3009 a formal hypothesis test pointless (Johnson, 1999). Certain aspects of SCR models are
3010 such an example. In SCR models, we will see that "spatial location" of individuals is
3011 an element of the model. The simpler, reduced, model is an ordinary capture-recapture
3012 model which is not spatially explicit (i.e., Chapt. 4), but it seems silly and pointless to
3013 think about actually using the reduced model even if we could concoct some statistical
3014 test to refute the more complex model. The simpler model is manifestly wrong but, more
3015 importantly, not even a plausible data-generating model! Other examples are when effort,
3016 area or sample rate is used as a covariate. One might prefer to have such things in models
3017 regardless of whether or not they pass some statistical litmus test.

3018 Many problems can be approached using one of these methods. In later chapters
3019 (especially Chapt. 8) we will address model selection in specific contexts and we hope
3020 those will prove useful for a majority of the situations you might encounter.

3.10 SUMMARY AND OUTLOOK

3021 GLMs and GLMMs are the most useful statistical methods in all of ecology. The prin-
3022 ciples and procedures underlying these methods are relevant to nearly all modeling and
3023 analysis problems in every branch of ecology. Therefore, understanding how to analyze
3024 these models is an essential skill for the quantitative ecologist to possess. If you under-
3025 stand and can conduct classical likelihood and Bayesian analysis of Poisson and binomial
3026 GL(M)Ms, then you will be successful analyzing and understanding more complex classes
3027 of models that arise. We will see shortly that spatial capture-recapture models are a
3028 type of GL(M)M and thus having a basic understanding of the conceptual origins and
3029 formulation of GL(M)Ms and their analysis is extremely useful.

3030 We note that GL(M)Ms are routinely analyzed by likelihood methods but we have
3031 focused on Bayesian analysis here in order to develop the tools that are less familiar
3032 to most ecologists, and that we will apply in much of the remainder of the book. In
3033 particular, Bayesian analysis of models with random effects is relatively straightforward
3034 because the models are easy to analyze conditional on the random effect, using MCMC.
3035 Thus, we will often analyze SCR models in later chapters by MCMC, explicitly adopting a
3036 Bayesian inference framework. In that regard, the various **BUGS** engines (**WinBUGS**,

OpenBUGS, **JAGS**; see also Appendix 1) are enormously useful because they provide an accessible platform for carrying out analyses by MCMC by just describing the model, and not having to worry about how to actually build MCMC algorithms. That said, the **BUGS** language is more important than just to the extent that it enables one to do MCMC - it is useful as a modeling tool because it fosters understanding, in the sense that it forces you to become intimate with your model. You have to think about and write down all of the probability assumptions, and the relationships between observations and latent variables and parameters in a way that is ecologically sensible and statistically coherent. Because of this, it focuses your thinking on *model construction*, as M. Kéry says in his **WinBUGS** book (Kéry, 2010), "**WinBUGS** frees the modeler in you."

While we have emphasized Bayesian analysis in this chapter, and make primary use of it through the book, we we will provide an introduction to likelihood analysis in Chapt. 6 and use those methods also from time to time. Before getting to that, however, it will be useful to talk about more basic, conventional closed population capture-recapture models and such models are the topic of the next chapter.