

1

2

Spatial Capture-Recapture Models

3

A hierarchical approach

4

The Four Horsemen (and women)

5

USGS Patuxent Wildlife Research Center
North Carolina State University

6

7

ACADEMIC PRESS

CONTENTS

8

9

10	1 Introduction	9
11	2 Introduction to Bayesian Analysis of GL(M)Ms Using R/WinBUGS	11
12		
13	2.1 Notation	12
14	2.2 GLMs and GLMMs	13
15	2.3 Bayesian Analysis	14
16	2.3.1 Bayes Rule	15
17	2.3.2 Bayesian Inference	16
18	2.3.3 Prior distributions	18
19	2.3.4 Posterior Inference	18
20	2.3.5 Small sample inference	19
21	2.4 Characterizing posterior distributions by MCMC simulation	20
22	2.5 What Goes on Under the MCMC Hood	21
23	2.5.1 Rules for constructing full conditional distributions	22
24	2.5.2 Metropolis-Hastings algorithm	23
25	2.6 Practical Bayesian Analysis and MCMC	24
26	2.6.1 Choice of prior distributions	24
27	2.6.2 Convergence and so-forth	25
28	2.6.3 Bayesian confidence intervals	26
29	2.6.4 Estimating functions of parameters	27
30	2.7 Bayesian Analysis using WinBUGS	27
31	2.7.1 Linear Regression in WinBUGS	27
32	2.7.2 Inference about functions of model parameters	30
33	2.8 Model Checking and Selection	30
34	2.8.1 Goodness-of-fit	31
35	2.8.2 Model Selection	32
36	2.9 Poisson GLMs	33
37	2.9.1 Important properties of the Poisson distribution	34
38	2.9.2 Example: Breeding Bird Survey Data	34
39	2.9.3 Doing it in WinBUGS	35
40	2.9.4 Constructing your own MCMC algorithm	38
41	2.10 Poisson GLM with Random Effects	43

42	2.11	Binomial GLMs	45
43	2.11.1	Binomial regression	45
44	2.11.2	Example: Waterfowl Banding Data	46
45	2.12	Summary and Outlook	48
46	3	Closed population models	51
47	4	Estimating the Size of a Closed Population	53
48	4.1	The Simplest Closed Population Model: Model M0	54
49	4.1.1	The Spatial Context of Capture-Recapture	56
50	4.1.2	Conditional likelihood	56
51	4.2	Data Augmentation	57
52	4.2.1	DA links occupancy models and closed population models . .	58
53	4.2.2	Model M_0 in BUGS	59
54	4.2.3	Formal development of data augmentation	61
55	4.2.4	Remarks on Data Augmentation	62
56	4.2.5	Example: Black Bear Study on Fort Drum	63
57	4.3	Temporally varying and behavioral effects	67
58	4.4	Models with individual heterogeneity	68
59	4.4.1	Analysis of Model Mh	71
60	4.4.2	Analysis of the Fort Drum data	72
61	4.4.3	Building your own MCMC algorithm	75
62	4.4.4	Exercises related to model Mh	77
63	4.5	Individual Covariate Models: Toward Spatial Capture-Recapture . .	78
64	4.5.1	Example: Location of capture as a covariate.	79
65	4.5.2	Extension of the Model	83
66	4.5.3	Invariance of density to maxD	86
67	4.5.4	Toward Fully Spatial Capture-recapture Models	86
68	4.6	DISTANCE SAMPLING: A primitive Spatial Capture-Recapture	
69		Model	87
70	4.6.1	Example: Muntjac deer survey from Nagarahole, India . . .	89
71	4.7	Summary and Outlook	90
72	5	Fully Spatial Capture-Recapture Models	93
73	6	Fully Spatial Capture-Recapture Models	95
74	6.1	Sampling Design and Data Structure	96
75	6.2	The binomial observation model	97
76	6.2.1	Distance as a latent variable	98
77	6.3	The Binomial Point-process Model	99
78	6.3.1	Definition of home range center	100
79	6.3.2	The state-space of the point process	101
80	6.3.3	Invariance and the State-space as a model assumption	102

81	6.3.4 Connection to Model Mh	103
82	6.3.5 Connection to Distance Sampling	104
83	6.4 Simulating SCR Data	104
84	6.4.1 Formatting and manipulating real data sets	106
85	6.5 Fitting an SCR Model in BUGS	106
86	6.6 Unknown N	109
87	6.6.1 Analysis using data augmentation in WinBUGS	110
88	6.6.2 Use of other BUGS engines: JAGS	113
89	6.7 Case Study: Wolverine Camera Trapping Study	114
90	6.7.1 Fitting the model in WinBUGS	117
91	6.7.2 Conclusion of Analysis	120
92	6.8 Constructing Density Maps	121
93	6.8.1 Example: Wolverine density map.	123
94	6.9 Discrete State-Space	123
95	6.9.1 Evaluation of Coarseness of Discrete Approximation	125
96	6.9.2 Analysis of the wolverine camera trapping data	127
97	6.9.3 SCR models as multi-state models	129
98	6.10 Summary and Outlook	129
99	7 Other observation models	133
100	8 Maximum likelihood estimation	135
101	9 Likelihood Analysis of SCR Models	137
102	9.1 Likelihood analysis	138
103	9.1.1 Implementation (simulated data)	139
104	9.2 MLE when N is Unknown	144
105	9.2.1 Exercises	146
106	9.2.2 Integrated Likelihood using the model under data augmenta- tion	147
107	9.2.3 Extensions	147
108	9.3 Classical model selection and assessment	147
109	9.4 Likelihood analysis of the wolverine camera trapping data	148
110	9.4.1 Exercises	150
111	9.5 Program DENSITY and the R package secr	151
112	9.5.1 Analysis using the secr package	152
113	9.6 Summary and Outlook	154
114		
115	10 MCMC details	157

116	11 MCMC Details	159
117	11.1 Introduction	159
118	11.1.1 Why build your own MCMC algorithm?	159
119	11.2 MCMC and posterior distributions	160
120	11.3 Types of MCMC sampling	163
121	11.3.1 Gibbs sampling	163
122	11.3.2 Metropolis-Hastings sampling	166
123	11.3.3 Metropolis-within-Gibbs	170
124	11.4 GLMMs Poisson regression with a random effect	170
125	11.4.1 Rejection sampling and slice sampling	174
126	11.5 MCMC for closed capture-recapture Model Mh	174
127	11.6 MCMC algorithm for the basic spatial capture-recapture model . . .	175
128	11.6.1 SCR model with binomial encounter process	178
129	11.6.2 Looking at model output	179
130	11.6.3 Posterior density plots	182
131	11.6.4 Serial autocorrelation and effective sample size	182
132	11.6.5 Summary results	184
133	11.6.6 Other useful commands	186
134	11.7 Manipulating the state-space	186
135	11.8 MCMC software packages	189
136	11.8.1 WinBUGS	189
137	11.8.2 OpenBUGS	189
138	11.8.3 JAGS Just Another Gibbs Sampler	190
139	11.9 Summary and Outlook	191
140	12 Goodness of Fit and stuff	193
141	13 Covariate models	195
142	14 State-space Covariates	197
143	14.1 Homogeneous point process revisited	198
144	14.2 Inhomogeneous binomial point process	199
145	14.3 Examples	201
146	14.3.1 Simulation and analysis of inhomogeneous point processes . .	201
147	14.3.2 Fitting inhomogeneous point process SCR models	204
148	14.3.3 The jaguar data	208
149	14.4 Summary	209
150	14.5 Other ideas	210
151	15 Inhomogeneous Point Process	211
152	16 Open models	213

154
155
156

1

INTRODUCTION

2

INTRODUCTION TO BAYESIAN ANALYSIS OF GL(M)MS USING R/WINBUGS

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. 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 GLM models - Poisson and binomial (i.e., logistic regression) GLMs - that will prove to be enormously useful in the analysis of capture-recapture models of all kinds. Many readers are probably familiar with these models because they represent probably the most generally useful models in all of 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**, which we will translate directly to the analysis of SCR models in subsequent chapters.

Bayesian analysis is convenient for analyzing GLMMs 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 in **WinBUGS** is, in part, the purpose of this chapter. While we use **WinBUGS** to do the Bayesian computations, we organize and summarize our data and execute

WinBUGS from within **R** using the useful package **R2WinBUGS** (Sturtz et al., 2005). Kéry (2010), and Kéry and Schaub (2011) provide excellent introductions to the basics of Bayesian analysis and GLMs at an accessible level. We don't want to be too redundant with those books and so we avoid a detailed treatment of Bayesian methodology - 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 (2009), and King (2009).

While this chapter is about Bayesian analysis of GLMMs, such models are routinely analyzed using likelihood methods too, as discussed by Royle and Dorazio (2008), and Kéry (2010). Indeed, likelihood analysis of such models is the primary focus of many applied statistics texts, a good one being Zuur et al. (2009). Later in this book, 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.

2.1 NOTATION

We will sometimes use conventional “bracket notation” to refer to probability distributions. If y is a random variable the $[y]$ indicates its distribution or its probability density/mass function (pdf, pmf) depending on context. If x is another random variable then $[y|x]$ is the conditional distribution of y given x , and $[y, x]$ is the joint distribution of y and x . To differentiate specific distributions in some contexts we might label them $g(y)$, $g(y|\theta)$, $f(x)$, or similar. We will also write $y \sim \text{Normal}(\mu, \sigma^2)$ to indicate that y “is distributed as” a normal random variable with parameters μ and σ^2 . The expected value or mean of a random variable is $E[y] = \mu$, and $\text{Var}[y] = \sigma^2$ is the variance of y . To indicate specific observations we'll use an index such as “ i ”. So, y_i for $i = 1, 2, \dots, n$ indicates observations for n individuals. Finally, we write $\text{Pr}(y)$ to indicate specific probabilities, i.e., of events “ y ” or similar.

To illustrate these concepts and notation, suppose z is a binary outcome (e.g., species occurrence) and we might assume the model: $z \sim \text{Bern}(p)$ for observations. Under this model $\text{Pr}(z = 1) = \psi$, which is also the expected value $E[z] = \psi$. The variance is $\text{Var}[z] = \psi * (1 - \psi)$ and the probability mass function (pmf) is $[z] = \psi^z (1 - \psi)^{1-z}$. Sometimes we write $[z|\psi]$ when it is important to emphasize the conditional dependence of z on ψ . As another example, suppose y is a random variable denoting whether or not a species is detected if an occupied site is surveyed. In this case it might be natural to express the pmf of the observations y *conditional* on z . That is, $[y|z]$. In this case, $[y|z = 1]$ is the conditional pmf of y given that a site is occupied, and it is natural to assume that $[y|z = 1] = \text{Bern}(p)$ where p is the “detection probability” - the probability that we detect the species, given that it is present. The model for the observations y is completely specified once we describe

the other conditional pmf $[y|z = 0]$. For this conditional distribution it is sometimes reasonable to assume $\Pr(y = 1|z = 0) = 0$ (MacKenzie et al. (2002); see also Royle and Link (2006)). That is, if the species is absent, the probability of detection is 0. This implies that $\Pr(y = 0|z = 0) = 1$. To allow for situations in which the true state z is unobserved, we assume that $[z]$ is Bernoulli with parameter ψ . In this case, the marginal distribution of y is

$$[y] = [y|z = 1]Pr(z = 1) + [y|z = 0]Pr(z = 0)$$

because $[y|z = 0]$ is a point mass at $y = 0$, by assumption, then

$$\Pr(y = 1) = p\psi$$

And

$$\Pr(y = 0) = (1 - p) * \psi + (1 - \psi)$$

2.2 GLMS AND GLMMS

We have asserted already that SCR models work out most of the time to be variations of GLMs and GLMMS. Some of you might therefore ask: What are GLMs and GLMMS, anyhow? These models are covered extensively in many very good applied statistics books and we refer the reader elsewhere for a detailed introduction. We think Kéry (2010), Kéry and Schaub (2011), and Zuur et al. (2009) are all accessible treatments of considerable merit. Here, we'll give the 1 minute treatment of GLMMS, not trying to be complete but rather only to preserve a coherent organization to the book.

The generalized linear model (GLM) is an extension of standard linear models by allowing the response variable to have some distribution from the exponential family of distributions (i.e., not just normal). This includes the normal distribution but also dozens of 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. Finally, GLMs typically accommodate a relationship between the mean and variance. The classical reference for GLMs is Nelder and Wedderburn (1972) and also McCullagh and Nelder (1989). The GLM consists of three components:

1. A probability distribution for the dependent variable y , from a class of probability distributions known as the exponential family.
2. A "linear predictor" $\eta = \mathbf{X}\beta$.
3. A link function g that relates $E[y]$ to the linear predictor, $E[y] = \mu = g^{-1}(\eta)$. Therefore $g(E[y]) = \eta$.

The dependent variable y is assumed to be an outcome from a distribution of the exponential family which includes many common distributions including the

normal, gamma, Poisson, binomial, and many others. The mean of the distribution of y is assumed to depend on predictor variables x according to

$$g(E[y]) = \mathbf{x}'\beta$$

where $E[y]$ is the expected value of y , and $\mathbf{x}'\beta$ is termed the *linear predictor*, i.e., a linear function of the predictor variables with unknown parameters β to be estimated. The function g is the link function. In standard GLMs, the variance of y is a function V of the mean of y : $Var(y) = V(\mu)$ (see below for examples).

A Poisson GLM posits that $y \sim \text{Poisson}(\lambda)$ with $E[y] = \lambda$ 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 $V(y_i) = \lambda_i$. The binomial GLM posits that $y_i \sim \text{Binomial}(K, p)$ where K is the fixed sample size parameter and $E[y_i] = K * p_i$. 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 $\text{logit}(u) = \log(u/(1-u))$. The inverse-logit function, g^{-1} , is a function we will refer to as “expit”, so that $\text{expit}(u) = \exp(u)/(1 + \exp(u))$.

A GLMM is the extension of GLMs to accommodate “random effects”. Often this involves adding a normal random effect to the linear predictor, and so a simple example is:

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

where

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

2.3 BAYESIAN ANALYSIS

Bayesian analysis is unfamiliar to many ecological researchers because older cohorts of ecologists were largely educated in the classical statistical paradigm of frequentist inference. But advances in technology and increasing exposure to 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 parameters¹ Instead, frequentists use probability to characterize the behavior of *procedures* such as estimators or confidence intervals (see below), which can lead to some inelegant or unnatural interpretations of things. It is paradoxical 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.

2.3.1 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:

$$[z|y] = [y|z][z]/[y]$$

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.

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 ($y = 0$), 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 presence 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 T times but never detect the species, then this clearly does not imply that the species is not present ($z = 0$) at this site. Rather, our degree of belief in $z = 0$ should be made with a probabilistic statement $\Pr(z = 1|y_1 = 0, \dots, y_T = 0)$. If the T surveys are independent so that we might regard y_t as *iid* Bernoulli trials, then the total number of detections, say y , is Binomial with probability p then we can use Bayes rule to compute the

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

probability that it is present given that it is not detected in T samples. In words, the expression we seek is:

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

Mathematically, this is

$$\begin{aligned} \Pr(z = 1|y = 0) &= \Pr(y = 0|z = 1) \Pr(z = 1) / \Pr(y = 0) \\ &= [(1 - p)^T \psi] / [(1 - p)^T \psi + (1 - \psi)]. \end{aligned}$$

To apply this, suppose that $T = 2$ surveys are done at a wetland for a species of frog, and the species is not detected there. Suppose further that $\psi = .8$ and $p = .5$ are obtained from a prior study. Then the probability that the species is present at this site is $.25 * .8 / (.25 * .8 + .2) = 0.50$. That is, there seems to be about 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 one needs to deduce one from the other. Bayes' rule as a basic fact of probability is not disputed.

2.3.2 Bayesian Inference

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). Of course, in Bayesian analysis they are also unknown and, in fact, there is a single data-generating value and so they are also fixed. The difference is that this fixed but unknown value is regarded as having been generated from some probability distribution. Specification of that probability distribution is necessary to carryout 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 $[\theta]$. 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 exist for characterizing the posterior distribution that do not require that we possess a mathematical understanding of $[y]$, i.e., we never have to compute it or know what it looks like, or know anything specific about it.

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. What is true is that, to a Bayesian, parameters are random variables. However, a Bayesian assumes, just like a frequentist, that there was a single data-generating value of that parameter - a fixed, and unknown value that produced the given data set. The distinction between Bayesian and frequentist approaches is that Bayesians regard the parameter as a random variable, and its value as the outcome of a random value, on par with the observations. This 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 - because parameters are not random variables!

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 which is mostly like² a “basket of methods” with little coherent organization. What is mostly coherent in frequentist inference is the manner in which items in this basket of methods 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 $E_y[\hat{\theta}|y]$ 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 fixed constants and estimators as random variables leads to interpretations that are not so straightforward. For example confidence

²Characterization from Sims REF XYZ

intervals having the interpretation “95% probability that the interval contains the true value” and p-values being “the probability of observing an outcome 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 the hypothetical realizations that characterize the performance of our procedure we will never get to observe.

While we do tend to favor Bayesian inference for the conceptual simplicity (parameters are random, posterior inference), we mostly advocate for a pragmatic non-partisian approach to inference because, frankly, some of these “bucket of methods” are actually very convenient in certain situations as we will see in later chapters.

2.3.3 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., if different investigators might report different results based on subjective assessments of things. Thus it is usually better to “let the data speak” and use priors that reflect absence of information beyond the data set being analyzed.

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 radius” and thus auxiliary information on the home range size of a species can be used as prior information (e.g., see Chandler and Royle (2012) ; also chapter XYZ).

XXXXXXXXX

noninformative prior on one scale is informative on another scale. e.g., flat prior on logit(p) is very different from uniform(0,1) on p... show graphic.....

reference to non-invariance of prior distributions to transformation.....

XXXXXXXXX

2.3.4 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 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 is between L and U is 0.95". It is not a subtle thing that this cannot be said using frequentist methods - although people tend to say it anyway and not really understand why it is wrong or even that it is wrong. This is actually a failing of frequentist ideas and the inability of frequentists to get people to overcome their natural tendency to use probability - which is something that, as a frequentist, you simply cannot do in the manner that you would like to.

Posterior inference is the main practical element of Bayesian analysis. We get to make an inference conditional on the data that we actually observed - i.e., what we actually know. To us, this seems logical - to condition on what we know. Conversely, frequentist inference is based on considering average performance over hypothetical unobserved data sets (i.e., the "relative frequency" interpretation of probability). Frequentists know that their procedures work well when averaged over all hypothetical, unobserved, data sets but no one ever really knows how well they work for the specific data set analyzed. That seems like a relevant question to biologists who oftentimes only have their one, extremely valuable, data set. This distinction comes into play a lot in exposing philosophical biases in the peer review of statistical analyses in ecology in the sense that, despite these opposing conceptual views to inference (i.e. conditional on the data you have, or averaged over hypothetical realizations), those who conduct a Bayesian analysis are often (in ecology, almost always) required to provide a frequentist evaluation of their Bayesian procedure.

2.3.5 Small sample inference

Using Bayesian inference, we obtain an estimate of the posterior distribution which 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). One of the great virtues of Bayesian analysis which is not really appreciated is that it is completely valid for any particular sample size. i.e., it is $[\theta|y]$, as precise as we claim it to be based on our ability to do calculations, for the particular sample size and observations that we have even if we have only a single datum y . The same cannot be said for almost all frequentist procedures in which estimates or variances are very often (almost always in practice) based on "asymptotic approximations" to the procedure which is actually 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 carrying out inference, such as calculating standard errors, doing model selection by 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. For example, a recent paper on the fossa (*Cryptoprocta ferox*), an endangered carnivore in Madagascar, estimated an adult density of 0.18 adults / km sq based on 20 animals captured over 3 years (Hawkins and Racey, 2005). A similar paper on the endangered southern river otter (*Lontra provocax*) estimated a density of 0.25 animals per river km based on 12 individuals captured over 3 years (Sepúlveda et al., 2007). Gardner et al. (2010) analyzed data from a study of the Pampas cat, a species for which very little is known, wherein only 22 individual cats were captured during the two year period. Trolle and Kéry (2005) reported only 9 individual ocelots captured and Jackson et al. (2006) captured 6 individual snow leopards using camera trapping. Thus, studies of rare and/or secretive carnivores necessarily and flagrantly violate one of Le Cam's Basic Principles, that of "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 datasets, and the resulting inference, are often criticized as being poor and unreliable. Or, even worse³, "the data set is so small, this is a poor analysis." On the other hand, such data 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, 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 some approximate $\Pr(D|data)$ deviates from the actual $\Pr(D|data)$ because they are precisely the same quantity.

2.4 CHARACTERIZING POSTERIOR DISTRIBUTIONS BY MCMC SIMULATION

In practice, it is not really feasible to ever compute the marginal probability distribution $\Pr(y)$, the denominator resulting from application of Bayes' rule. For decades 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 stuff 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,

³Actual quote from a referee

514 have to devise various approximations to what they were trying to characterize.
 515 The advent of Markov chain Monte Carlo (MCMC) methods has made it easier to
 516 calculate posterior distributions for just about any problem to arbitrary levels of
 517 precision.

518 Broadly speaking, MCMC is a class of methods for drawing random numbers
 519 (sampling or simulating) from the target posterior distribution. Thus, even though
 520 we might not recognize the posterior as a named distribution or be able to ana-
 521 lyze its features analytically, e.g., devise mathematical expressions for the mean
 522 and variance, we can use these MCMC methods to obtain a large sample from the
 523 posterior and then use that sample to characterize features of the posterior. What
 524 we do with the sample depends on our intentions – typically we obtain the mean or
 525 median for use as a point estimate, and take a confidence interval based on Monte
 526 Carlo estimates of the quantiles. These are estimates, but not like frequentist es-
 527 timates. Rather, they are Monte Carlo estimates with an associated Monte Carlo
 528 error which is largely determined arbitrarily by the analyst. They are not estimates
 529 qualified by a sampling distribution as in classical statistics. If we run our MCMC
 530 long enough then our reported value of $E[\theta|y]$ or any feature of the posterior dis-
 531 tribution is precisely what we say it is. There is no “sampling variation” in the
 532 frequentist sense of the word. In summary, the MCMC samples provide a Monte
 533 Carlo characterization of *the* posterior distribution.

2.5 WHAT GOES ON UNDER THE MCMC HOOD

534 We will develop and apply MCMC methods in some detail for spatial capture-
 535 recapture models in chapter 10. Here we provide a simple illustration of some basic
 536 ideas related to the practice of MCMC.

537 A type of MCMC method relevant to most problems is Gibbs sampling (REF
 538 XYZ XYZ), which is based on the idea of iterative simulation from the “full con-
 539 ditional” distributions (also called conditional posterior distributions). The full
 540 conditional distribution for an unknown quantity is the conditional distribution of
 541 that quantity given every other random variable in the model - the data and all other
 542 parameters. For example, for a normal regression model with $y \sim \text{Normal}(\alpha + \beta x, 1)$
 543 then the two full conditionals are, in symbolic terms,

$$[\alpha|y, \beta]$$

544 and

$$[\beta|y, \alpha].$$

545 We might use our knowledge of probability to identify these mathematically. In
 546 particular, by Bayes’ Rule, $[\alpha|y, \beta] = [y|\alpha, \beta][\alpha|\beta]/[y|\beta]$ and similarly for $[\beta|y, \alpha]$.
 547 For example, if we have priors for $[\alpha]$ and $[\beta]$ which are also normal distributions,
 548 some algebra reveals that XXXX COPY NOTATION FFROM CH. 6 XXXXX

$$[\alpha|y, \beta] = \text{Normal}(\text{ybar}, \dots \text{weightedvariancehere} \dots).$$

549 Similarly,

$$[\beta|y, \alpha] \text{isnormal}(\dots\dots)$$

550 The MCMC algorithm for this model has us simulate in succession, repeat-
 551 edly, from those two distributions. See Gelman et al. (2004) for more examples of
 552 Gibbs sampling for the normal model. A conceptual representation of the MCMC
 553 algorithm for this simple model is therefore: XXXX Check out ALGORITHM en-
 554 vironment XXXXX

555 **Algorithm**

```
556
557     0. Initialize  $\alpha$  and  $\beta$ 
558
559     Repeat{
560         1. Draw a new value of  $\alpha$  from Eq. \ref{xyz}
561
562         2. Draw a new value of  $\beta$  from Eq. \ref{xyz}
563     }
```

564 As we just saw for this simple “normal-normal” model it is sometimes possible
 565 to specify the full conditional distributions analytically. In general, when certain
 566 so-called conjugate prior distributions are chosen, the form of full conditional distri-
 567 butions is similar to that of the observation model. In this normal-normal case, the
 568 normal distribution for the mean parameters is the conjugate prior under the normal
 569 model, and thus the full-conditional distributions are also normal. This is conven-
 570 nient because, in such cases, we can simulate directly from them using standard
 571 methods (or **R** functions). But, in practice, we don’t really ever need to know such
 572 things because most of the time we can get by using a simple algorithm, called the
 573 Metropolis-Hastings (henceforth “MH”) algorithm, to obtain samples from these
 574 full conditional distributions without having to recognize them as specific, named,
 575 distributions. This gives us enormous freedom in developing models and analyzing
 576 them without having to resolve them mathematically because to implement the MH
 577 algorithm we need only identify the full conditional distribution up to a constant
 578 of proportionality, that being the marginal distribution in the denominator (e.g.,
 579 $[y|\beta]$ above).

580 We will talk about the Metropolis-Hastings algorithm shortly, and we will use
 581 it extensively in the analysis of SCR models (e.g., chapter 10).

582 2.5.1 Rules for constructing full conditional distributions

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

586 (step 1) Collect all stochastic components of the model;

- 587 (step 2) Recognize and express the full conditional in question as proportional to
 588 the product of all components;
 589 (step 3) Remove the ones that don't have the focal parameter in them.
 590 (step 4) Do some algebra on the result in order to identify the resulting pdf or pmf.

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

596 To illustrate for computing $[\alpha|y, \beta]$ we first apply step 1 and identify the model
 597 components as: $[y|\alpha, \beta]$, $[\alpha]$ and $[\beta]$. Step 2 has us write $[\alpha|y, \beta] \propto [y|\alpha, \beta][\alpha][\beta]$.
 598 Step 3: We note that $[\beta]$ is not a function of alpha and therefore we remove it to
 599 obtain $[\alpha|y, \beta] \propto [y|\alpha, \beta][\alpha]$. Similarly we obtain $[\beta|y, \alpha] \propto [y|\alpha, \beta][\beta]$. We apply
 600 step 4 and manipulate these algebraically to arrive at the result or, alternatively, we
 601 can sample them indirectly using the Metropolis-Hastings algorithm (see below).

602 2.5.2 Metropolis-Hastings algorithm

603 The Metropolis-Hastings (MH) algorithm is a completely generic method for sam-
 604 pling from any distribution, say $f(\theta)$. In our applications, $f(\theta)$ will typically be
 605 the full conditional distribution of θ . While we sometimes use Gibbs sampling,
 606 we seldom use “pure” Gibbs sampling because we might use MH to sample from
 607 one or more of the full conditional distributions. When the MH algorithm is used
 608 to sample from full conditional distributions of a Gibbs sampler the resulting hy-
 609 brid algorithm is called *Metropolized Gibbs sampling* or more commonly *Metropolis-*
 610 *within-Gibbs*. Shortly we will actually construct such an algorithm for a simple
 611 class of models.

612 The MH algorithm generates candidates from some proposal or candidate-
 613 generating distribution, that may be conditional on the current value of the pa-
 614 rameter, denoted by $h(\theta^*|\theta^t)$. Here, θ^* is the *candidate* or proposed value and θ^t is
 615 the current value, i.e., at iteration t of the MCMC algorithm. The proposed value
 616 is accepted with probability XXXX check notation with Rahel XXXXXX

$$r = \frac{f(\theta^*)h(\theta^t|\theta^*)}{f(\theta^t)h(\theta^*|\theta^t)}$$

617 which we call the MH acceptance probability. This ratio can sometimes be > 1 in
 618 which case we set it equal to 1. It is useful to note that $h()$ can be anything at all.
 619 Absolutely anything! You can generate candidate values from a *normal*(0,1) dis-
 620 tribution, from a *uniform*(-3455,3455) distribution, or anything of proper support.
 621 Note, however, that good choices of $h()$ are those that approximate the posterior
 622 distribution. Obviously if $h() = f(\theta|y)$ (i.e., the posterior) then you always accept
 623 the draw, and it stands to reason that proposals that are more similar to $f(\theta|y)$

will lead to higher acceptance probabilities. No matter the choice of $h()$, we can evaluate this ratio numerically because the marginal $f(y)$ cancels from both the numerator and denominator, which is the magic of the MH algorithm.

A special kind of $h()$ are those that are symmetric, which means that $h(a|b) = h(b|a)$ in which case $h(a|b)$ and $h(b|a)$ just cancel out from the MH acceptance probability and r is then just the ratio of the target density evaluated at the candidate value to that evaluated at the current value. A type of symmetric proposal useful in many situations is the so-called *random-walk* proposal distribution where candidate values are drawn from a normal distribution with mean equal to the current value and some standard deviation, say δ , which is prescribed by the user. For parameters that have support on the real line, e.g., α in our example above, the random walk proposal generator has us generate $\alpha^* \sim \text{Normal}(\alpha^t, \delta)$. If we set δ very small we have a high probability of accepting the proposal and vice versa. In practice, we “tune” delta to achieve a compromise between acceptance rate and efficient mixing of the Markov chains (see below for an example) normally assessed by autocorrelation. Low δ increases the acceptance rate but will tend to produce Markov chains with high autocorrelation, and vice versa.

Parameters with bounded support: Many models contain parameters that have bounded support. E.g., variance parameters live on $[0, \infty]$, parameters that represent probabilities live on $[0, 1]$, etc.. In that case it is sometimes convenient to use a random walk proposal distribution that can generate any real number (e.g., a normal random walk proposal). In that case, we can just reject parameters that are outside of the parameter space (XXXX REF FOR THIS XXXX).

2.6 PRACTICAL BAYESIAN ANALYSIS AND MCMC

There are a number of really important practical issues to be considered in any Bayesian analysis and we cover some of these briefly here.

2.6.1 Choice of prior distributions

**XXX integrate this material with previous section on prior distributions
XXXXXX**

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 $\text{Unif}(a, b)$ 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; (2) “improper” uniform priors or (3) sometimes even a bounded $\text{Unif}(a, b)$ 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 $\text{Norm}(0, .01)$ will typically

be used. Of course $\tau = 0.01$ ($\sigma^2 = 100$) might be very informative for a regression parameter that has a high variance. Therefore, we recommend that predictor variables *always* be standardized. 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

$$\text{logit}(p_i) = \alpha + \beta x_i$$

Would be $[\alpha] = \text{const}$ which is the same as saying $a \sim \text{Unif}(\infty, \text{inf})$, the customary improper uniform prior. However, we might also use a prior on the parameter $p_0 = \text{logit}^{-1}(a)$, which is $\text{Pr}(y = 1)$ for the value $x = 0$. Since p_0 is a probability a natural choice is $p_0 \sim \text{Unif}(0, 1)$. These two priors can affect results (see Chapter 3.XYZ), yet they are both sensible non-informative priors. 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. XXX REFS on prior distributions XXXXX

2.6.2 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 of the most important is “have the chains converged?” Most MCMC algorithms only guarantee that, eventually, the samples being generated will be from the target posterior distribution. So-called “convergence” of the Markov chain is achieved when that happens. 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.

The quick diagnostic to whether convergence has been achieved is that your Markov chains look “grassy” – see Fig. 2.5 below. Another way to check convergence is to update the parameters some more and see if the posterior changes. It is good to confirm convergence using the “R-hat” statistic (\hat{R}) or Brooks-Gelman-Rubin statistic (Gelman et al., 1996) which should be close to 1 if the Markov chains have converged and sufficient posterior samples have been obtained. In practice, $\hat{R} = 1.2$ is probably 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 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. So, for example, using model selection methods (section XYZ) sometimes suggests non-convergence. Another situation is when one of the parameters is on the boundary of the parameter space which might appear to be very poor mixing, but all within some extreme region of the parameter space.⁴ This

⁴it would be nice if we could compile examples of this later in the book and reference back to this point

kind of stuff is 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 what people might also say “have not covered” (or “slow convergence”) which is a term we would disagree with because the samples might well be from the posterior (i.e., the Markov chains have converged to the proper stationary distribution) but simply mix around the posterior rather slowly. Anyway, poor mixing can happen for a huge number of reasons – when parameters are highly correlated (even confounded), or barely identified from the data, or the algorithms are very terrible and probably many other reasons. 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 to a tolerable level. A strategy often used to reduce autocorrelation is “thinning” – i.e., keep every m^{th} value of the Markov chain output. 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). 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. MCMC is an art, and a science.

Is the posterior sample large enough? A good rule of thumb is that you should never report MCMC results to more than 2 decimal places – because they will always be different! Look at the MC error which is printed by default in summaries of BUGS output. You want that to be smallish relative to the magnitude of the parameter and this might 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, but you can run your MCMC algorithm as long as it takes. Note that MC error in summaries of the posterior is not the same as having an “approximate” solution in a standard likelihood analysis or similar. The approximate SE in likelihood inference is actually wrong in its actual value.... XYZ.

2.6.3 Bayesian confidence intervals

The 95% Bayesian interval based on percentiles of the posterior is not a unique interval – there are many of them – and the so-called “highest posterior density” (HPD) interval is the narrowest interval. We might compute that frequently because

it is easy to do with an integer parameter which N is (See the next chapter). The 95 % HPD is not often exactly 95% but usually slightly more conservative than nominal because it is the narrowest interval that contains at least 95% of the posterior mass.

2.6.4 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 $i = 1, 2, \dots, M$ 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 $i = 1, 2, \dots, M$. We give another example in section 2.7.2 below and throughout this book. 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 chapter 5).

2.7 BAYESIAN ANALYSIS USING WINBUGS

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** (REF XYZ XYZ) or **rjags** packages. **WinBUGS** and **JAGS** are MCMC black boxes that takes a pseudo-code description (i.e., written in the **BUGS** language) of all of the relevant stochastic and deterministic elements of a model and generates an MCMC algorithm for that model. But you never get to see the algorithm. Instead, **WinBUGS/JAGS** will run the algorithm and just 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** or **JAGS** in this book, we note that **OpenBUGS** is the current active development tree of the **BUGS** language. See Kéry (2010, ch.xyz) and Kéry and Schaub (2011, appendix xyz) for more on practical analysis in **WinBUGS**. That book should also be consulted for a more comprehensive introduction to using **WinBUGS**. In this example, we're going to accelerate pretty fast.

2.7.1 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, the command line being indicated by ‘>’. First, simulate a covariate x and observations y having prescribed intercept, slope and variance:

```
777 > x<-rnorm(10)
778 > mu<- -3.2+ 1.5*x
781 > 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`:

```
785 > cat("
786 model {
787   for (i in 1:10){
788     y[i]~dnorm(mu[i],tau)      # the "likelihood"
789     mu[i]<- beta0 + beta1*x[i] # the linear predictor
790   }
791   beta0~dnorm(0,.01)          # prior distributions
792   beta1~dnorm(0,.01)
793   sigma~dunif(0,100)
794   tau<-1/(sigma*sigma)        # tau is a derived parameter
795 }
796 ",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.

Remarks: 1. WinBUGS parameterizes the normal in terms of the mean and inverse-variance, called the precision. Thus, `dnorm(0,.01)` implies a variance of 100; **2.** 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 $+B$. **3.** We typically use a `Unif(0, B)` prior on standard deviation parameters (Gelman XXX 2006 XXXX). But sometimes we might use a gamma prior on the precision parameter τ . **4.** In a **WinBUGS** 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 “~”, or it has to be a derived parameter connected to variables and data using “<-”.

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 **R** workspace and also in the **WinBUGS** model definition. We also have to create an **R** function that produces a list of starting values `inits` that get sent to **WinBUGS**. 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. All of these activities look like this:

```
library("R2WinBUGS") # "attach" the R2WinBUGS library
data <- list ( "y","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=2, n.chains=2,
           n.burnin=2000, n.iter=6000, debug=TRUE,working.dir=getwd())
```

Remarks: A common question is “how should my data be formatted?” That depends on how you describe the model in the **BUGS** language, how your data are input into **R** and subsequently formatted. 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 chapter 5 and elsewhere. In general, starting values are optional but we recommend to always provide reasonable starting values for structural parameters, but are not always necessary for random effects. 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 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).

You should execute all of the commands given above and then look at the resulting output. Kill the **WinBUGS** GUI and the data will be read back into **R** (or specify `debug=FALSE`). We don’t want to give instructions on how to navigate and use the GUI - see XYZ REF (XYZ) for that. The object `out` prints important summaries by default (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), n.thin = 2
n.sims = 4000 iterations saved
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
beta0	-2.43	1.84	-6.21	-3.50	-2.42	-1.34	1.27	1	4000
beta1	2.62	1.54	-0.42	1.68	2.62	3.57	5.67	1	4000
sigma	5.29	1.66	3.11	4.14	4.95	6.05	9.39	1	4000
tau	0.05	0.02	0.01	0.03	0.04	0.06	0.10	1	4000
deviance	59.85	3.24	56.18	57.47	59.00	61.37	68.32	1	840

861 For each parameter, `n.eff` is a crude measure of effective sample size,
 862 and `Rhat` is the potential scale reduction factor (at convergence, `Rhat=1`).
 863
 864 DIC info (using the rule, `pD = Dbar-Dhat`)
 865 `pD = 2.6` and `DIC = 62.4`
 866

867 **Remarks:** (1) convergence is assessed using the \hat{R} statistic – which we might
 868 sometimes write “*Rhat*”. A value of *Rhat* near 1 indicates convergence; (2) DIC
 869 is the “deviance information criterion” (Spiegelhalter et al., 2002) (see section 2.8)
 870 which some people use in a manner similar to AIC although it is recognized to
 871 have some problems in hierarchical models (Millar, 2009). We evaluate this in the
 872 context of SCR models in chapter XYZ XYZ.

873 2.7.2 Inference about functions of model parameters

874 Using the MCMC draws for a given model we can easily obtain the posterior distri-
 875 bution of any function of model parameters. We showed this in the above example
 876 by providing the posterior of τ when the model was parameterized in terms of stan-
 877 dar deviation σ . As another example, suppose that the normal regression model
 878 above had a quadratic response function of the form

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

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

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

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

2.8 MODEL CHECKING AND SELECTION

886 In general terms model checking - or assessing the adequacy of the model - and
 887 model selection are quite thorny issues and, despite contrary and, sometimes,
 888 strongly held belief among practitioners, there are not really definitive, general
 889 solutions to either problem. We're against dogma on these issues and think people
 890 need to be open-minded about such things and recognize that models can be useful
 891 whether or not they pass certain statistical tests. Some models are intrinsically
 892 better than others because they make more biological sense or foster understanding
 893 or achieve some objective that some bootstrap or other goodness-of-fit test can't

894 decide for you. That said, it gives you some confidence if your model seems ade-
 895 quate and we try to provide some fit assessment in most real applications of SCR
 896 models We provide a very brief overview of concepts here, but provide more detailed
 897 coverage in chapter 12. See also Kéry (2010, ch. xyz) and Link and Barker (2009,
 898 ch. xyz) for specific context related to Bayesian model checking and selection.

899 2.8.1 Goodness-of-fit

900 Goodness-of-fit testing is an important element of any analysis because our model
 901 represents a general set of hypotheses about the ecological and observation pro-
 902 cesses that generated our data. Thus, if our model “fits” in some statistical or
 903 scientific sense, then we believe it to be consistent with the hypotheses that went
 904 into the model. More formally, we would conclude that the data are *not inconsis-*
 905 *tent* with the hypotheses, or that the model appears adequate. If we have enough
 906 data, then of course we will reject any set of statistical hypotheses. Conversely,
 907 we can always come up with a model that fits by making the model extremely
 908 complex. Despite this paradox, it seems to us that simple models that you can
 909 understand should usually be preferred even if they don’t fit, for example if they
 910 embody essential mechanisms central to our understanding of things, or if we think
 911 that some contributing factors to lack-of-fit are minor or irrelevant to the scientific
 912 context and intended use of the model. In other words, models can be useful irre-
 913 spective of whether they fit according to some formal statistical test of fit. Yet the
 914 tension is there to obtain fitting models, and this comes naturally at the expense of
 915 models that can be easily interpreted and studied and effectively used. Moreover,
 916 conducting goodness-of-fit tests is not always so easy to do. Moreover, it is never
 917 really easy (or especially convenient) to decide if your goodness-of-fit test is worth
 918 anything. It might have 0 power! Despite this, we recommend attempting to assess
 919 model fit in real applications, as a general rule, and we provide some basic guidance
 920 here and some more specific to SCR models in chapter 12.

921 To evaluate goodness-of-fit in Bayesian analyses, we will most often use the
 922 Bayesian p-value (Gelman et al., 1996). The basic idea is to define a fit statistic or
 923 “discrepancy measure” and compare the posterior distribution of that statistic to
 924 the posterior predictive distribution of that statistic for hypothetical perfect data
 925 sets for which the model is known to be correct. For example, with count frequency
 926 data, a standard measure of fit is the sum of squares of the “Pearson residuals”,

$$D(y_i, \theta) = \frac{(y_i - E(y_i))^2}{\text{Var}(y_i)}$$

927 The fit statistic based on the squared residuals is

$$FIT = \sum_i D(y_i, \theta)^2$$

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, simulated using the current parameter values. The Bayesian p-value is simply the posterior probability $\Pr(\text{Fit} > \text{Fit}_{\text{new}})$ ⁵ 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 . In summary, the Bayesian p-value seems like a bootstrap idea, is easy to compute, and widely used as a result.

Another useful fit statistic is the Freeman-Tukey statistic⁶, in which

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

(Brooks et al., 2000), where y_i is the observed value of observation i and e_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.

2.8.2 Model Selection

For model selection we typically use three different methods: First is, let’s say, common sense. If a parameter 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 with all of the interest over the last 10 or 15 years on model selection in ecology. It seems reasonable to us.

For regression problems we sometimes use the factor weighting idea which is to introduce a set of binary variables w_k for variable k , and express the model as, e.g., for a single covariate model:

$$E(y_i) = \alpha + w\beta x_i$$

where w is given a Bernoulli prior distribution with some prescribed probability. E.g., $w \sim \text{Bern}(0.50)$ 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 $w = 1$ is a gauge of the importance of the variable x . i.e., high values of $\Pr(w = 1)$ indicate stronger evidence to support that “ x is in the model” whereas values of $\Pr(w = 1)$ close to 0 suggest that x is less important.

This idea seems to be due to Kuo and Mallick (1998)⁷ and see Royle and Dorazio (2008, ch. XXXX) for an example in the context of logistic regression. This

⁵Check this definition!

⁶Ref for this?

⁷ Is this also what people call Zellner’s G-priors?

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 with this approach is that its effectiveness and results will typically be highly sensitive to the prior distribution on the structural parameters (e.g., see Royle and Dorazio (2008, table xyz)). The reason for this is obvious: If $w = 0$ for the current iteration of the MCMC algorithm, so that β is sampled from the prior distribution, and the prior distribution is very diffuse, then extreme values of β are likely. Consequently, when the current value of β is far away from the mass of the posterior when $w = 1$, then the Markov chain may only jump from $w = 0$ to $w = 1$ infrequently. One seemingly reasonable solution to this problem (Aitken XYZ FIND THIS XXXXX⁸) is to fit the full model to obtain posterior distributions for all parameters, and then use those as prior distributions in a “model selection” run of the MCMC algorithm. This seems preferable to more-or-less arbitrary restriction of the prior support to improve the performance of the MCMC algorithm.

A third method that we advocate is subject-matter context. It seems that there are some situations – some models – where one should not have to do model selection because it is necessitated by the specific context of the problem, thus rendering a formal hypothesis test pointless (Johnson, 1999). SCR models are such an example. In SCR models, we will see that “spatial location” of individuals is an element of the model. The simpler, reduced, model is an ordinary capture-recapture model which is not spatially explicit (i.e., chapter 11), but it seems silly and pointless to think about actually using the reduced model even if we could concoct some statistical test to refute the more complex model. The simpler model is manifestly wrong but, more importantly, not even a plausible data-generating model! Other examples are when effort, area or sample rate is used as a covariate. One might prefer to have such things in models regardless of whether or not they pass some statistical litmus test (although one can always find referees to argue for pedantic procedure over thinking).

Many problems can be approached using one of these methods but there are also broad classes of problems that can’t and, for those, you’re on your own. In later chapters we will address model selection in specific contexts and we hope those will prove useful for a majority of the situations you encounter.

2.9 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 $y_i; i = 1, 2, \dots, n$ follow a Poisson distribution with mean λ which we write

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

⁸see Royle 2008 paper for reference

Commonly y_i is a count of animals or plants at some point in space and λ_i might depend on i . For example, i might index point count locations in a forest, BBS route centers, or sample quadrats, or similar. 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,

$$\log(x(i)) = \alpha + \beta * x(i)$$

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, $\text{Var}(y) = \text{E}(y) = \lambda$. Thus the model accommodates a linear increase in variance with the mean.

2.9.1 Important properties of the Poisson distribution

There are two properties of the Poisson distribution that make it extremely useful in ecology. First is the property of *compound additivity*. If y_1 and y_2 are Poisson random variables with means λ_1 and λ_2 , then their sum $N = y_1 + y_2$ is Poisson with mean $\lambda_1 + \lambda_2$. Thus, if the observations can be viewed as an aggregate of counts over some finer unit of measurement, then the mean aggregates in a corresponding manner. Secondly, the Poisson distribution has a direct relationship to the multinomial. If y_1 and y_2 are *iid* Poisson then, conditional on their sum $N = y_1 + y_2$, their joint distribution is multinomial with sample size N and cell probabilities $\lambda_1/(\lambda_1 + \lambda_2)$ and $\lambda_2/(\lambda_1 + \lambda_2)$. As a result of this, most multinomial models can be analyzed as a Poisson GLM and *vice versa*.

2.9.2 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 counts made along North American Breeding Bird Survey (BBS) routes in Pennsylvania, USA. A route consists of 50 stops separated by 0.5 mile. 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 have is 1966-1998. BBS data can be obtained online at <http://www.pwrc.usgs.gov/bbs/>. 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. For 1990 there were 77 active routes. We have the data stored in a .csv file⁹ where rows 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 route was not run are coded as “NA” in the data matrix. We imagine that this will be a typical format for many ecological

⁹check this data format

1033 studies, perhaps with more columns representing covariates. To read in the data
1034 and display the first few elements of this matrix, do this:

```
1035 > a<-read.csv("pa-bbsdovedata-all.csv")
1036 > data[1:2,1:6]
1037      X      lon      lat      habitat X66 X67
1038 1 72002 -80.445 41.501 -0.3871372 NA 24
1039 2 72003 -80.347 41.214 -1.0171629 NA NA
```

1040 It is useful to display the spatial pattern in the observed counts. For that we
1041 use a spatial dot plot - where we plot the coordinates of the observations and mark
1042 the color of the plotting symbol based on the magnitude of the count. We have a
1043 special plotting function for that which is called `spatial.plot()` and it is available
1044 with the supplemental **R** package. Actually, what we want to do here is plot the
1045 log-count (+1 of course) which (Fig. 2.1) displays a notable pattern that could be
1046 related to something. The **R** commands for obtaining this figure are:

```
1047 data<-read.csv("pa-bbsdovedata-all.csv")
1048 y<-data[,29] # pick out 1990
1049 notna<-!is.na(y)
1050 y<-y[notna]
1051 spatial.plot(data[notna,2:3],y)
```

1052 We can ponder the potential effects that might lead to dove counts being
1053 high....corn fields, telephone wires, barn roofs along with misidentification of pi-
1054 geons, these could all correlated reasonably well with the observed count of mourn-
1055 ing doves. Unfortunately we don't have any of that information.

1056 We do have a measure of forest cover in the vicinity of each point which is
1057 contained in the data set (variable "habitat"). This was derived from a larger GIS
1058 coverage of the state (provided in the data file "pahabdata.csv") which can be
1059 plotted using the `spatial.plot` function using the following commands

```
1060 > map('state',regions="penn",lwd=2)
1061 > spatial.plot(pahabdata[,2:3],pahabdata[, "dfor"],cx=2)
1062 > map('state',regions="penn",lwd=2,add=TRUE)
```

1063 where the result appears in Fig. 2.2. We see a prominent pattern that indicates
1064 high forest coverage in the central part of the state and low forest cover in the SE.
1065 Inspecting the previous figure of log-counts suggests a relationship between counts
1066 and forest cover which is perhaps not surprising.

1067 2.9.3 Doing it in WinBUGS

1068 Here we demonstrate how to fit a Poisson GLM in **WinBUGS** using the covariate
1069 x_i = forest cover. It is advisable that x_i be standardized in most cases as this will
1070 improve mixing of the Markov chains. Recall that the data we have stored include
1071 a standardized covariate (forest cover) and so we don't have to worry about that
1072 here. To read the BBS data into **R** and get things set up for **WinBUGS** we issue
1073 the following commands:

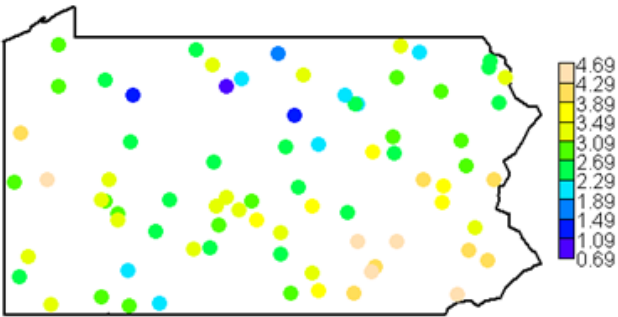


Figure 2.1. Needs a caption

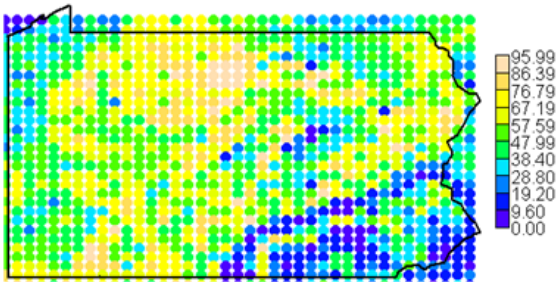


Figure 2.2. Needs a caption

```

1074 data<-read.csv("pa-bbsdovedata-all.csv")
1075 y<-data[,29] # pick out 1990
1076 notna<-!is.na(y)
1077 y<-y[notna] # discard missing
1078 habitat<-data[notna,4] # get habitat data
1079 library("R2WinBUGS") # load R2WinBUGS
1080 data <- list ( "y","M","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**:

```

1084 cat("
1085 model {
1086     for (i in 1:M){
1087         y[i]~dpois(lam[i])
1088         log(lam[i])<- beta0+beta1*habitat[i]
1089     }
1090     beta0~dunif(-5,5)
1091     beta1~dunif(-5,5)
1092 }
1093 ",file="PoissonGLM.txt")
1094
1095 inits <- function() list ( beta0=rnorm(1),beta1=rnorm(1))
1096 parameters <- c("beta0","beta1")
1097 out<-bugs (data, inits, parameters, "PoissonGLM.txt", n.thin=2,n.chains=2,
1098           n.burnin=2000,n.iter=6000,debug=TRUE,working.dir=getwd())

```

Remarks: (1) Note the close correspondence in how the model is specified here compared with the normal regression model previously. As an exercise you should discuss the specific differences between the **BUGS** model specifications for the normal and Poisson models.

```

1103 > print(out,digits=3)
1104 Inference for Bugs model at
1105 ‘‘PoissonGLM.txt’’, fit using WinBUGS,
1106 2 chains, each with 4000 iterations (first 1000 discarded), n.thin = 2
1107 n.sims = 3000 iterations saved

```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
beta0	3.151	0.025	3.102	3.135	3.151	3.168	3.199	1.001	2300
beta1	-0.498	0.021	-0.539	-0.512	-0.498	-0.484	-0.457	1.001	3000
fit	869.930	19.856	835.500	855.700	868.600	881.900	913.602	1.002	1600
fitnew	76.709	12.519	54.098	68.107	76.215	84.510	102.602	1.001	3000
deviance	1116.605	2.014	1115.000	1115.000	1116.000	1117.000	1122.000	1.001	3000

We might wonder whether this model provides an adequate fit to our data. To evaluate that, we used a Bayesian p-value analysis with fit statistic based on the Freeman-Tukey residual by replacing the model specification above with this:

```

1117 cat("
1118 model {
1119     for (i in 1:M){
1120         y[i]~dpois(lam[i])
1121         log(lam[i])<- beta0+beta1*habitat[i]
1122         d[i]<- pow(pow(y[i],0.5)-pow(lam[i],0.5),2)    #
1123
1124         ynew[i]~dpois(lam[i])
1125         dnew[i]<-pow( pow(ynew[i],0.5)-pow(lam[i],0.5),2)
1126
1127     }
1128     fit<-sum(d[])
1129     fitnew<-sum(dnew[])
1130     beta0~dunif(-5,5)
1131     beta1~dunif(-5,5)
1132 }
1133 ",file="PoissonGLM.txt")

```

1134 The Bayesian p-value is the proportion of times $fitnew > fit$ which, for this
 1135 data set, is 0, which was 1.0 in this case (calculation omitted). This suggests that
 1136 the basic Poisson model does not fit well.

1137 2.9.4 Constructing your own MCMC algorithm

1138 It might be helpful to suffer through a couple examples building custom MCMC
 1139 algorithms. Here, we develop an MCMC algorithm for the Poisson regression model,
 1140 using a Metropolis-within-Gibbs sampling framework.

1141 We will assume that the two parameters have diffuse normal priors, say $[\alpha] =$
 1142 $\text{Norm}(0, 100)$ and $[\beta] = \text{Norm}(0, 100)$ where each has *standard deviation* 100 (recall
 1143 that **WinBUGS** parameterizes the normal in terms of $1/\sigma^2$). We need to assemble
 1144 the relevant elements of the model which are these two prior distributions and the
 1145 likelihood $[\mathbf{y}|\alpha, \beta] = \prod_i [y_i|\alpha, \beta]$ which is, mathematically, the product of the Poisson
 1146 pmf evaluated at each y_i , given particular values of α and β . Next, we need to
 1147 identify the full conditionals $[\alpha|\beta, \mathbf{y}]$ and $[\beta|\alpha, \mathbf{y}]$. We use the all-purpose rule for
 1148 constructing full conditionals (section 2.5.1) to discover that:

$$[\alpha|\beta, \mathbf{y}] \propto \left\{ \prod_i [y_i|\alpha, \beta] \right\} [\alpha]$$

1149 and

$$[\beta|\alpha, \mathbf{y}] \propto \left\{ \prod_i [y_i|\alpha, \beta] \right\} [\beta]$$

1150 Remember, we could replace the “ \propto ” with “ $=$ ” if we put $[y|\beta]$ or $[y|\alpha]$ in the de-
 1151 nominator. But, in general, $[y|\alpha]$ or $[y|\beta]$ will be quite a pain to compute and, more

importantly, it is a constant as far as the operative parameters (α or β , 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 draw, and so the denominator required to change \propto to $=$ winds up canceling from the MH acceptance probability. Here we will use the random walk candidate generator so that, for example, $\alpha^* \sim \text{Normal}(\alpha^t, \delta)$ where δ is the standard-deviation of the proposal distribution, which is just a tuning parameter¹⁰. 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, $a*b = \exp(\log(a) + \log(b))$. The “Metropolis within Gibbs” algorithm for a Poisson regression turns out to be remarkably simple:

```

1163 set.seed(2013)
1164
1165 out<-matrix(NA,nrow=1000,ncol=2)  # matrix to store the output
1166 alpha<- -1                        # starting values
1167 beta <- -.8
1168
1169 # begin the MCMC loop ; do 1000 iterations
1170 for(i in 1:1000){
1171
1172   # update the alpha parameter
1173   lambda<- exp(alpha+beta*habitat)
1174   lik.curr<- sum(log(dpois(y,lambda)))
1175   prior.curr<- log(dnorm(alpha,0,100))
1176   alpha.cand<-rnorm(1,alpha,.25)      # generate candidate
1177   lambda.cand<- exp(alpha.cand + beta*habitat)
1178   lik.cand<- sum(log(dpois(y,lambda.cand)))
1179   prior.cand<- log(dnorm(alpha.cand,0,100))
1180   mhratio<- exp(lik.cand +prior.cand - lik.curr-prior.curr)
1181   if(runif(1)< mhratio)
1182     alpha<-alpha.cand
1183
1184   # update the beta parameter
1185   lik.curr<- sum(log(dpois(y,exp(alpha+beta*habitat))))
1186   prior.curr<- log(dnorm(beta,0,100))
1187   beta.cand<-rnorm(1,beta,.25)
1188   lambda.cand<- exp(alpha+beta.cand*habitat)
1189   lik.cand<- sum(log(dpois(y,lambda.cand)))
1190   prior.cand<- log(dnorm(beta.cand,0,100))
1191   mhratio<- exp(lik.cand + prior.cand - lik.curr - prior.curr)
1192   if(runif(1)< mhratio)
1193     beta<-beta.cand
1194

```

¹⁰ It would help lots of people out to see a non-symmetric proposal distribution, and the extra step needed to account for it.

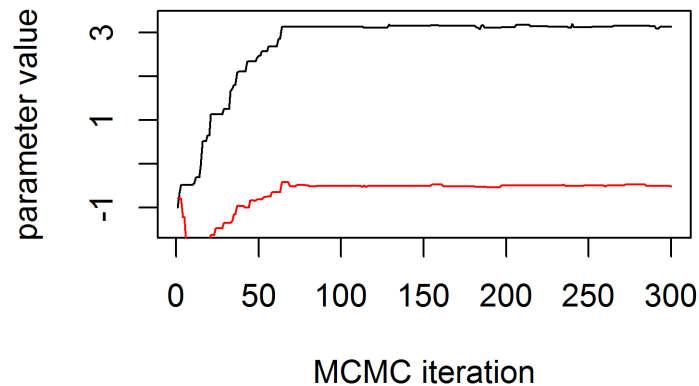


Figure 2.3. MCMC output for Poisson regression parameters (top trace: intercept α ; bottom trace: slope β). This is for $\delta = 0.25$.

```

1195 out[i,]<-c(alpha,beta)          # save the current values
1196 }
1197
1198
1199 plot(out[,1],ylim=c(-1.5,3.3),type="l",lwd=2,ylab="parameter value",
1200      xlab="MCMC iteration")
1201 lines(out[,2],lwd=2,col="red")

```

The first 300 iterations of the MCMC history of each parameter is shown in Fig. 2.3. The appearance of this is not very appealing but a couple of things are evident: First, the Markov chains clearly stabilize - “burn-in” - after about 60 or 70 iterations. They also appear to mix very slowly once convergence is achieved, although this is not so clear given the scale of the y -axis. We decreased the standard deviation of the candidate generating distribution from $\delta = 0.25$ to $\delta = 0.05$ and re-ran the MCMC algorithm producing the output shown in Fig. 2.4. We see that the burn-in takes longer but it seems to mix better although it takes slightly longer to burn-in. Using this value of δ we generated 10,000 posterior samples, discarding the first 500 as burn-in, and the result is shown in Fig. 2.5, this time 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

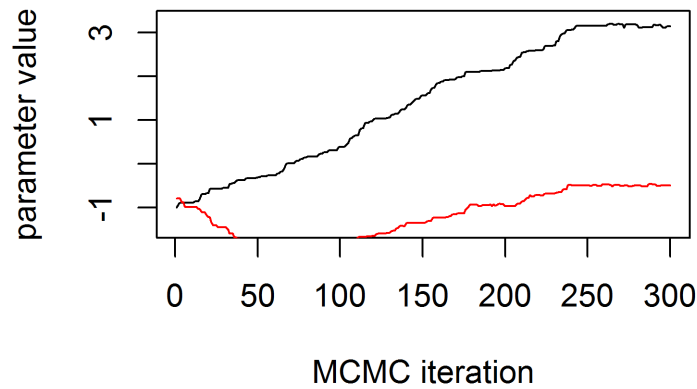


Figure 2.4. Same as previous fig but with $\delta = 0.05$.

1214 satisfied with results that look like this.

1215 **Remarks:** We used a specific set of starting values for these simulations. It
 1216 should be clear that starting values closer to the mass of the posterior distribution
 1217 might cause burn-in to occur faster. As an exercise, evaluate that. (2) Clearly
 1218 the influence of the proposal standard deviation term is important. Small values
 1219 lead to much better mixing but it should be noted that values that are too small
 1220 will slow down burn-in and also lead to high correlation. This suggests there is an
 1221 optimal value of the Metropolis-Hastings tuning parameter¹¹. As an exercise you
 1222 should contemplate finding that optimal value for this problem¹² (3) For the flat
 1223 normal prior distributions here we could leave the prior contribution out of the full
 1224 conditional evaluation since it is locally constant, i.e., constant in the vicinity of the
 1225 posterior mass, and thus has no practical effect. Removing the prior contribution
 1226 from the MH acceptance probability is equivalent to saying that the parameters
 1227 have an improper uniform prior, i.e., $\alpha \sim \text{const}$, which is commonly used for mean
 1228 parameters in practice. Note also that we have used a different prior than in our
 1229 **WinBUGS** model specification given previously. As an exercise, evaluate whether
 1230 this seems to affect the result.

¹¹Defined previously?????

¹²effective sample size definition?

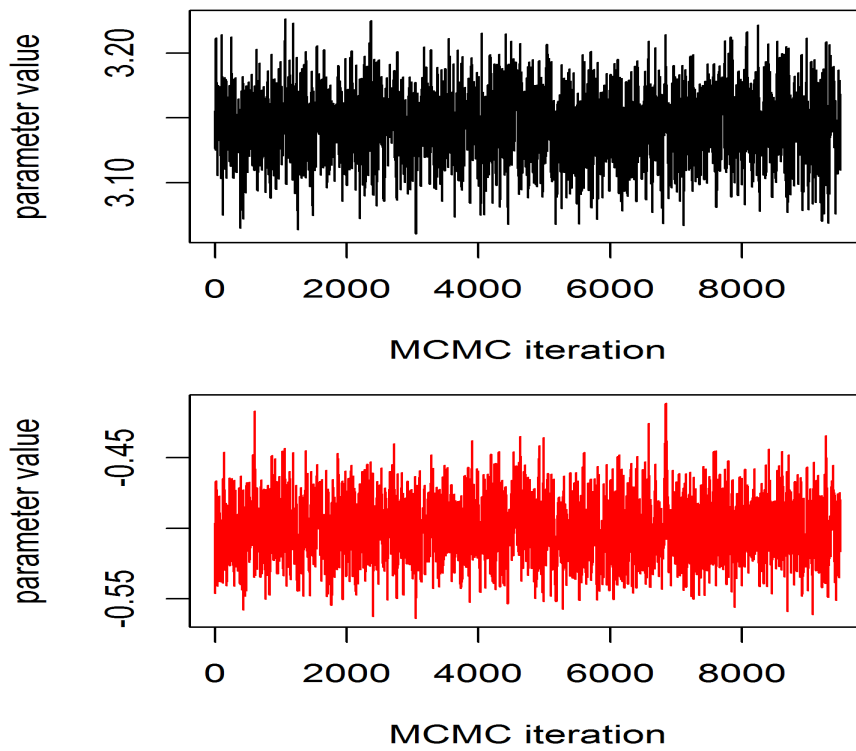


Figure 2.5. nice grassy mcmc output, longer run of previous with $\delta = 0.05$.

2.10 POISSON GLM WITH RANDOM EFFECTS

What we will be doing in most of this book is 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 by way of example, extending our Poisson regression model to include a random effect.

ANDY STOPPED HERE

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:

$$\log(\lambda_i) = \alpha + \beta x_i + \eta_i$$

where $\eta_i \sim \text{Normal}(0, \sigma^2)$. A natural alternative is to have multiplicative gamma-distributed noise, $\exp(\eta_i) \sim \text{Gamma}(a, b)$ which would correspond to a negative binomial kind of over-dispersion, implying a different mean/variance relationship to the log-normal mixture (the interested reader should work that out). Choosing between such possibilities is not a topic we will get into here because it doesn't seem possible to provide general guidance on it. For this model we carried-out a goodness-of-fit evaluation using the Bayesian p-value based on a Pearson residual statistic. See also (Kéry, 2010, ch. 18) for an example involving a binomial mixed model¹³. Anyhow, it is really amazingly simple to express this model in **WinBUGS** and have **WinBUGS** draw samples from the posterior distribution using the following code for the BBS dove counts:

```
data<-read.csv("pa-bbsdovedata-all.csv")
locs<-data[,2:3]
habitat<-data[,4]
y<-data[,29]      # grab year 1990
M<-length(y)

set.seed(2013)

cat("
model {
  for (i in 1:M){
    y[i]~dpois(lam[i])
    log(lam[i])<- alpha+ beta*habitat[i] + eta[i]
    frog[i]<-beta*habitat[i] + eta[i]
    eta[i] ~ dnorm(0,tau)
    d[i]<- pow(pow(y[i],0.5)-pow(lam[i],0.5),2)

    ynew[i]~dpois(lam[i])
    dnew[i]<- pow(pow(ynew[i],0.5)-pow(lam[i],0.5),2)
  }
  fit<-sum(d[])
```

¹³Kéry has noticed that such tests probably have 0 power. Should use the marginal frequency of the data

```

1271 fitnew<-sum(dnew[])
1272
1273 alpha~dunif(-5,5)
1274 beta~dunif(-5,5)
1275 sigma~dunif(0,10)
1276 tau<-1/(sigma*sigma)
1277 }
1278
1279 ",file="model.txt")
1280 data <- list ( "y","M","habitat")
1281 inits <- function()
1282   list ( alpha=rnorm(1),beta=rnorm(1),sigma=runif(1,0,4))
1283 parameters <- c("alpha","beta","sigma","tau","fit","fitnew")
1284 library("R2WinBUGS")
1285
1286 out<-bugs (data, inits, parameters, "model.txt", n.thin=2,n.chains=2,
1287   n.burnin=1000,n.iter=5000,debug=TRUE)

```

1288 This produces the following posterior summary statistics:

```

1289 > print(out,digits=2)
1290 Inference for Bugs model at "model.txt", fit using WinBUGS,
1291 2 chains, each with 5000 iterations (first 1000 discarded), n.thin = 2
1292 n.sims = 4000 iterations saved
1293
1294      mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
1295 alpha    2.98 0.08  2.82  2.93  2.98  3.03  3.12 1.00  1400
1296 beta   -0.53 0.07 -0.68 -0.58 -0.53 -0.49 -0.38 1.01   350
1297 sigma    0.60 0.06  0.49  0.56  0.59  0.64  0.73 1.00  2000
1298 tau     2.88 0.57  1.88  2.47  2.86  3.24  4.12 1.00  2000
1299 fit     26.58 3.72 19.87 23.96 26.37 29.01 34.46 1.00  4000
1300 fitnew   26.83 3.90 19.60 24.12 26.68 29.36 35.04 1.00  4000
1301 deviance 445.94 12.18 424.00 437.40 445.20 453.90 471.50 1.00  4000
1302
1303 [... some output deleted ...]

```

1303 The Bayesian p-value for this model is

```

1304 > mean(out$sims.list$fit>out$sims.list$fitnew)
1305 [1] 0.4815

```

1306 indicating a pretty good fit. Given the site-level random effect, it would be surpris-
 1307 ing for this model to not fit! One thing we notice is that the posterior standard
 1308 deviations of the regression parameters are much higher, a result of the excess vari-
 1309 ation. Wwe would also notice much less precise predictions of hypothetical new
 1310 observations.

1311 ANDY STOPPED HERE.

2.11 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 $E(y) = pK$. 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. 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, 2004a; 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)¹⁴. 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 the chapter 11.

2.11.1 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 i . The standard choice is the “logit” link function which is:

$$\log(p_i/(1 - p_i)) = \alpha + \beta * x_i.$$

¹⁴Some 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”

1344 The inverse-logit (or “expit”) is

$$p_i = \text{expit}(\alpha + \beta * x_i) = \frac{\exp(\alpha + \beta * x_i)}{1 + \exp(\alpha + \beta * x_i)}$$

1345 There are many other possible link functions. However, ecologists seem to adopt
 1346 the logit link function without question in most applications¹⁵. We sometimes use
 1347 the “complementary log-log” (= “cloglog”) link function in ecological applications
 1348 because it arises naturally in many situations (Royle and Dorazio, 2008, p. 150).
 1349 For example, consider the “probability of observing a count greater than 0” under
 1350 a Poisson model: $\Pr(y > 0) = 1 - \exp(-\lambda)$. In that case,

$$\text{cloglog}(p) = \log(-\log(1 - p)) = \log(\lambda)$$

1351 So that if you have covariates in your linear predictor for $E(y)$ under a Poisson
 1352 model then they are linear on the complementary log-log link of p . In models of
 1353 species occurrence it seems natural to view occupancy as being derived from local
 1354 abundance N (Royle and Nichols, 2003; Royle and Dorazio, 2006; Dorazio, 2007).
 1355 Therefore, models of local abundance in which $N \sim \text{Poisson}(A\lambda)$ for a habitat patch
 1356 of area A implies a model for occupancy ψ of the form

$$\text{cloglog}(\psi) = \log(A) + \log(\lambda).$$

1357 We will use the cloglog link in some analyses of SCR models in chapter 5 and
 1358 elsewhere.

1359 2.11.2 Example: Waterfowl Banding Data

1360 It would be easy to consider a standard “distribution modeling” application where
 1361 $K = 1$ and the outcome is occurrence ($y = 1$) or not ($y = 0$) of some species.
 1362 Such examples abound in books (e.g., Royle and Dorazio (2008, ch. 3); Kéry (2010,
 1363 ch. 21); Kéry and Schaub (2011, ch. 13)) and in the literature. Instead, we will
 1364 consider an example involving band returns of waterfowl which were analyzed by
 1365 Royle and Dubovsky (2001)¹⁶.

1366 For these data, y_i is the number of waterfowl bands recovered out of B_i birds
 1367 banded at some location \mathbf{s}_i . In this case B_i is fixed. Thinking about recovery rate
 1368 as being proportional to harvest rate, we use these data to explore geographic gra-
 1369 dients in recovery rate resulting from variability in harvest pressure experienced by
 1370 populations depending on their migration ecology. As such, we fit a basic binomial
 1371 GLM with a linear response to geographic coordinates (including an interaction
 1372 term). The data are provided with the **R** package **scrbook**. Here we provide the
 1373 part of the script for creating the model and fitting the model in **WinBUGS** using

¹⁵a notable exception is distance sampling, which is all about choosing among link functions

¹⁶I hate this example. Anyone got a better one thats not distribution modeling?

the `bugs` function. 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 main distributional assumption to binomial (specified with `dbin`) and then use the `logit` function to relate the parameter p_{it} to the covariates. Here is the script:

```

1379 load("mallarddata") # not sure how this will look
1380
1381 sink("model.txt")
1382 cat("
1383 model {
1384   for(t in 1:5){
1385     for (i in 1:nobs){
1386       y[i,t] ~ dbin(p[i,t], B[i,t])
1387       logit(p[i,t]) <- alpha0[t] + alpha1*X[i,1] + alpha2*X[i,2] + alpha3*X[i,1]*X[i,2]
1388     }
1389   }
1390   alpha1~dnorm(0,.001)
1391   alpha2~dnorm(0,.001)
1392   alpha3~dnorm(0,.001)
1393   for(t in 1:5){
1394     alpha0[t] ~ dnorm(0,.001)
1395   }
1396 }
1397 ",fill=TRUE)
1398 sink()
1399
1400 data <- list(B=mallard.bandings, y=mallard.recoveries,
1401             nobs=nrow(banding.locs),X=banding.locs)
1402 inits <- function(){
1403   list(alpha0=rnorm(5),alpha1=0,alpha2=0,alpha3=0) }
1404 parms <- list('alpha0','alpha1','alpha2','alpha3')
1405 out <- bugs(data,inits, parms,"model.txt",n.chains=3,
1406            n.iter=2000,n.burnin=1000, n.thin=2,debug=TRUE)

```

Posterior summaries of model parameters are as follows:

```

1408 > print(out,digits=3)
1409 Inference for Bugs model at "model.txt", fit using WinBUGS,
1410 3 chains, each with 2000 iterations (first 1000 discarded), n.thin = 2
1411 n.sims = 1500 iterations saved

```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
alpha0[1]	-2.346	0.036	-2.417	-2.370	-2.346	-2.323	-2.277	1.001	1500
alpha0[2]	-2.356	0.032	-2.420	-2.379	-2.356	-2.335	-2.292	1.001	1500
alpha0[3]	-2.220	0.035	-2.291	-2.244	-2.219	-2.197	-2.153	1.001	1500
alpha0[4]	-2.144	0.039	-2.225	-2.169	-2.143	-2.116	-2.068	1.000	1500
alpha0[5]	-1.925	0.034	-1.990	-1.949	-1.924	-1.901	-1.856	1.004	570

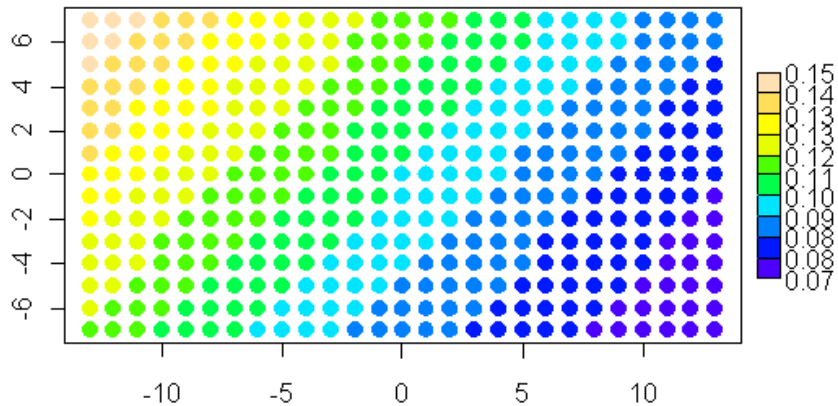


Figure 2.6. Predicted recovery rate of bands.

```
1418 alpha1      -0.023 0.003   -0.028  -0.025  -0.023  -0.022  -0.018  1.001  1500
1419 alpha2       0.020 0.006    0.009   0.016   0.020   0.024   0.031  1.001  1500
1420 alpha3       0.000 0.001   -0.002  -0.001   0.000   0.000   0.002  1.001  1500
1421 deviance  1716.001 4.091 1710.000 1713.000 1715.000 1718.000 1726.000 1.001  1500
1422
1423 [... some output deleted ...]
```

1424 The basic result suggests a negative east-west gradient and a positive south to
1425 north gradient but no interaction. A map of the response surface is shown in Fig.
1426 2.6. We did an additional MCMC run where we saved the binomial parameter
1427 p and computed the Bayesian p-value (double use of “p” here is confusing, but I
1428 guess that happens sometimes!) using a fit statistic based on the Freeman-Tukey
1429 statistic (see Section XXX above). The result indicates that the linear response
1430 surface model does not provide an adequate fit of the data. The reader should
1431 contemplate whether this invalidates the basic interpretation of the result.

2.12 SUMMARY AND OUTLOOK

1432 GLMs and GLMMs are the most useful statistical methods in all of ecology. The
1433 principles and procedures underlying these methods are relevant to nearly all mod-
1434 eling and analysis problems in every branch of ecology. Moreover, understanding
1435 how to analyze these models is crucial in a huge number of diverse problems. If you

1436 understand and can conduct classical likelihood and Bayesian analysis of Poisson
1437 and binomial GLM(M)s, then you will be successful analyzing and understanding
1438 more complex classes of models that arise. We will see shortly that spatial capture-
1439 recapture models are a type of GLMM and thus having a basic understanding of
1440 the conceptual origins and formulation of GLM(M)s and their analysis is extremely
1441 useful.

1442 We note that GLM(M)s are routinely analyzed by likelihood methods but we
1443 have focused on Bayesian analysis here in order to develop the tools that are less
1444 familiar to most ecologists. In particular, Bayesian analysis of models with ran-
1445 dom effects is relatively straightforward because the models are easy to analyze
1446 conditional on the random effect, using methods of MCMC. Thus, we will often
1447 analyze SCR models in later chapters by MCMC, explicitly adopting a Bayesian
1448 inference framework. In that regard, the various **BUGS** engines (**WinBUGS**,
1449 **OpenBUGS**, **JAGS**) are enormously useful because they provide an accessible
1450 platform for carrying out analyses by MCMC by just describing the model, and not
1451 having to worry about how to actually build MCMC algorithms. That said, the
1452 **BUGS** language is more important than just to the extent that it enables one to
1453 do MCMC - it is useful as a modeling tool because it fosters understanding, in the
1454 sense that it forces you to become intimate with your model. You have to write
1455 down all of the probability assumptions, the relationships between observations and
1456 latent variables and parameters. This is really a great learning paradigm that you
1457 can grow with.

1458 While we have emphasized Bayesian analysis in this chapter, and make primary
1459 use of it through the book, we will provide an introduction to likelihood analysis
1460 in chapter 9 and use those methods also from time to time. Before getting to that,
1461 however, it will be useful to talk about more basic, conventional closed population
1462 capture-recapture models and these are the topic of the next chapter.

3

1463

1464

1465

CLOSED POPULATION MODELS

4

ESTIMATING THE SIZE OF A CLOSED POPULATION

In this chapter we will consider ordinary capture-recapture (CR) models for estimating population size in closed populations. We will see that such models are closely related to binomial (or logistic) regression type models. In fact, when N is known, they are precisely such models. We consider some important extensions of ordinary closed population models that accommodate various types of “individual effects” — either in the form of explicit covariates (sex, age, body mass) or unstructured “heterogeneity” in the form of an individual random effect. In general, these models are variations of generalized linear or generalized linear mixed models (GLMMs). Because of the paramount importance of this concept, we focus mainly on fairly simple models in which the observations are individual encounter frequencies, y_i = the number of encounters of individual i out of K replicate samples of the population which, for the models we consider here, is the outcome of a binomial random variable. Along the way, we consider the spatial context of capture-recapture data and models and demonstrate that density cannot be formally estimated when spatial information is ignored. We also review some of the informal methods of estimating density using CR methods, and consider some of their limitations. We will be exposed to our first primitive spatial capture-recapture models which arise as relatively minor variations of so-called “individual covariate models” (of the Huggins (1989) and Alho (1990) variety). In a sense, the point of this chapter is to establish that linkage in a direct and concise manner beginning with the basic “Model M0” and extensions of that model to include individual heterogeneity and also individual covariates. A special type of individual covariate models is distance sampling, which could be thought of as the most primitive spatial capture-recapture model. In later chapters we further develop and extend

ideas introduced in this chapter.

We emphasize Bayesian analysis of capture-recapture models and we accomplish this using a method related to classical “data augmentation” from the statistics literature Tanner and Wong (e.g., 1987)). This is a general concept in statistics but, in the context of capture-recapture models where N is unknown, it has a consistent implementation across classes of capture-recapture models and one that is really convenient from the standpoint of doing MCMC (Royle et al., 2007). We use data augmentation throughout this book and thus emphasize its conceptual and technical origins and demonstrate applications to closed population models. We refer the reader to Kery and Schaub (2011, ch. 6) for an accessible and complimentary development of ordinary closed population models.

4.1 THE SIMPLEST CLOSED POPULATION MODEL: MODEL M0

We suppose that there exists a population of N individuals which we subject to repeated sampling, say over K nights, where individuals are captured, marked, and subsequently recaptured. We suppose that individual encounter histories are obtained, and these are of the form of a sequence of 0’s and 1’s indicating capture ($y = 1$) or not ($y = 0$) during any sampling occasion (“sample”). As an example, suppose $K = 5$ sampling occasions, then an individual captured during sample 2 and 3 but not otherwise would have an encounter history of the form $\mathbf{y} = (0, 1, 1, 0, 0)$. Thus, the observation \mathbf{y}_i for each individual (i) is a vector having elements denoted by y_{ik} for $k = 1, 2, \dots, K$. Usually this is organized as a row of a matrix with elements y_{ik} , see Table 4.1. Except where noted explicitly, we suppose that observations are independent within individuals and among individuals. Formally, this allows us to say that y_{ik} are Bernoulli random variables and we may write $y_{ik} \sim \text{Bern}(p)$. Consequently, for this very simple model in which p is in fact constant, then we can declare that the individual encounter frequencies (total captures), $y_i = \sum_k y_{ik}$, have a binomial distribution based on a sample of size K . That is

$$y_i = \sum_k y_{ik} \sim \text{Bin}(p, K)$$

for every individual in the population. This is a remarkably simple model that forms the cornerstone of almost all of classical capture-recapture models, including most spatial capture-recapture models discussed throughout this book. Evidently, the basic capture-recapture model structure is precisely a simplistic version of a logistic-regression model with only an intercept term ($\text{logit}(p) = \text{constant}$). To say that all capture-recapture models are just logistic regressions is only slightly inaccurate. In fact, we are proceeding here “conditional on N ”, i.e., as if we knew N . In practice we don’t, of course, and that is kind of the point of capture-recapture models as estimating N is the central objective. But, by proceeding conditional on N , we can specify a simple model and then deal with the fact that N is unknown

Table 4.1. a capture-recapture data set with $n = 6$ observed individuals and $K = 5$ samples.

indiv i	Sample occasion					y_i
	1	2	3	4	5	
1	1	0	0	1	0	2
2	0	1	0	0	1	2
3	1	0	0	1	0	2
4	1	0	1	0	1	3
5	0	1	0	0	0	1
$n = 6$	1	0	0	0	0	1

using standard methods that you are already familiar with (i.e., GLMs - see chapter 2).

Assuming individuals of the population are observed independently, the joint probability distribution of the observations is the product of N binomials

$$\begin{aligned} \Pr(y_1, \dots, y_N | p) &= \prod_{i=1}^N \text{Bin}(y_i | K, p) \\ &= \prod_{k=0}^K \pi(k)^{n_k} \end{aligned}$$

where $\pi(k) = \text{Bin}(k | K, p)$ and where $n_k = \sum_{i=1}^N I(y_i = k)$ denotes the number of individuals captured k times in K surveys. We emphasize that this is conditional on N , in which case we get to observe the $y = 0$ observations and the resulting data are just *iid* binomial counts. Because this is a binomial regression model of the variety described in chapter 2, fitting this model using a BUGS engine poses no difficulty.

The essential problem in capture-recapture, however, is that N is not known because the number of uncaptured/missing individuals (i.e., those in the zero cell that occur with probability $\pi(0)$) is unknown. Consequently, the observed capture frequencies n_k are no longer independent. Instead, their joint distribution is multinomial (e.g., see Illian (2008a, p. xyz)):

$$n_1, n_2, \dots, n_K \sim \text{Multin}(N, \pi(1), \pi(2), \dots, \pi(K)) \quad (4.1.1)$$

Note that in our notation the number of uncaptured/missing individuals is denoted by $n_0 = N - n$, where $n = \sum_{k=1}^K n_k$ denotes the total number of distinct individuals seen in the K samples.

To fit the model in which N is *unknown*, we can regard N as a parameter and maximize the multinomial likelihood directly. While direct likelihood analysis of the multinomial model is straightforward, that does not prove to be too useful in practice because we seldom are concerned with models for the aggregated encounter history frequencies. In many instances, including for spatial capture-recapture (SCR)

models, we require a formulation of the model that can accommodate individual level covariates which we address subsequently in this chapter.

4.1.1 The Spatial Context of Capture-Recapture

A common assumption made is that of population “closure” which is really just a colloquial way of saying (in part) the Bernoulli assumptions stated explicitly above. In the biological context, closure means, strictly, no additions or subtractions from the population during study. This is manifest by the statement that the encounters are independent and identically distributed (iid) Bernoulli trials. In practice, closure is usually interpreted by the manner in which potential violations of that assumption arise. In particular, two important elements of the closure assumption are “demographic” and “geographic” closure. If an individual dies then subsequent values of y_{ik} are clearly no longer Bernoulli trials with the same parameter p . If there is no mortality or recruitment in the population, then we say that demographic closure is satisfied. Similarly, animals may emigrate or immigrate. If they do not, then geographic closure is satisfied. Sometimes a distinction is made between temporary and permanent emigration or immigration. That is a relevant distinction in spatial capture-recapture models, because SCR models explicitly accommodate “temporary emigration” of a certain type, due to individuals moving about their home range. The demographic closure assumption can also be relaxed using SCR models, but we will save that discussion for chapter 5.

4.1.2 Conditional likelihood

We saw that a basic closed population model is a simple logistic regression model if N is known and, when N is unknown, the model is multinomial with index or sample size parameter N . This multinomial model, being conditional on N , is sometimes referred to as the “joint likelihood” the “full likelihood” or the “unconditional likelihood” (or model in place of likelihood). This formulation differs from the so-called “conditional likelihood” approach in which the likelihood of the observed encounter histories is devised conditional on the event that an individual is captured at least once. To construct this likelihood, we have to recognize that individuals appear or not in the sample based on the value of the random variable y_i , that is, we capture them if and only if $y_i > 0$. The observation model is therefore based on $\Pr(y|y > 0)$. For the simple case of Model M0, the resulting conditional distribution is a “zero truncated” binomial distribution which accounts for the fact that we cannot observe the value $y = 0$ in the data set (see Royle and Dorazio, 2008, section XYZ). Both the conditional or unconditional models are legitimate modes of analysis in all capture-recapture types of studies, and they provide equally valid descriptions of the data and for many practical purposes provide equivalent inferences, at least in large sample sizes (Sanathanan, 1972).

Mode of analysis	parameters in model	statistical model
Joint likelihood	p, N	multinomial with index N
Conditional likelihood	p	zero-truncated binomial
Data augmentation	p, ψ	zero-inflated binomial

Table 4.2. Modes of analysis of capture-recapture models.

In this book we emphasize Bayesian analysis of capture-recapture models using data augmentation (discussed subsequently), which produces yet a third distinct formulation of capture-recapture-models based on the zero-*inflated* binomial distribution that we describe in the next section. Thus, there are 3 distinct formulations of the model – or models of analysis – for analyzing all capture-recapture models based on the (1) binomial model for the joint or unconditional specification; (2) zero-truncated binomial that arises “conditional on n ”; and (3) the zero-inflated binomial that arises under data augmentation. Each formulation has a distinct complement of model parameters (shown in Table 4.2 for Model M0).

4.2 DATA AUGMENTATION

We consider a method of analyzing closed population models using data augmentation (DA) which is useful for Bayesian analysis and, in particular, analysis of models using the various BUGS engines and other software. Data augmentation is a general statistical concept that is widely used in statistics in many different settings. The classical reference is Tanner and Wong (1987) but see also Liu and Wu (1999). Data augmentation can be adapted to provide a very generic framework for Bayesian analysis of capture-recapture models with unknown N . This idea was introduced for closed populations by Royle et al. (2007), and has subsequently been applied to a number of different contexts including individual covariate models (Royle, 2009), open population models (Royle and Dorazio, 2008, 2010; Gardner et al., 2010), spatial capture-recapture models (Royle and Young, 2008; Royle, 2010; Gardner, 2009), and many others.

Conceptually, data augmentation takes the data you wish you had - that is, the data set with N rows - the known- N data set - and embeds that data set into a larger data set having $M > N$ rows.¹ It is always possible, in practice, to choose M pretty easily for a given problem and context. Then, under data augmentation, analysis is focused on the “augmented data set.” That is, we analyze the bigger data set - the one having M rows - with an appropriate model that accounts for the augmentation. Inference is focused directly on estimating the proportion $\psi = E[N]/M$, instead of directly on N , where ψ is the “data augmentation parameter.”

¹ RC: Might be just me, but I find that formulation a little confusing... I think it's the 'data you wish you had because that's effectively data you don't have. I think it might be easier to grasp if this were explained with the data you do have - based on n .

4.2.1 DA links occupancy models and closed population models

We provide a heuristic description of data augmentation based on the close correspondence between so-called “occupancy” models and closed population models following Royle and Dorazio (2008, sec. xyz).

In occupancy models (MacKenzie et al., 2002; Tyre et al., 2003) the sampling situation is that M sites, or patches, are sampled multiple times to assess whether a species occurs at each site. This yields encounter data such as that illustrated in the left panel of Table 4.3. The important problem is that a species may occur at a site, but go undetected, yielding the “all-zero” encounter histories which are observed. However, some of the all-zeros may well correspond to sites where the species in fact *does not* occur. Thus, while the zeros are observed, there are too many of them and, in a sense, the inference problem is to allocate the zeros into “structural” (fixed) and “sampling” (or stochastic) zeros. More formally, inference is focused on the parameter ψ , the probability that a site is occupied. In contrast, in classical closed population studies, we observe a data set as in the middle panel of Table 4.3 where *no* zeros are observed. The inference problem is, essentially, to estimate how many sampling zeros there are - or should be - in a “complete” data set. The inference objective (how many sampling zeros?) is precisely the same for both types of problems if an upper limit M is specified for the closed population model. The only distinction being that, in occupancy models, M is set by design (i.e., the number of sites to visit) whereas a natural choice of M for capture-recapture models may not be obvious. However, by assuming a uniform prior for N on the integers $[0, M]$, this upper bound is induced (Royle et al., 2007). Then, one can analyze capture-recapture models by adding $M - n$ all-zero encounter histories to the data set and regarding the augmented data set, essentially, as a site-occupancy data set.

Thus, the heuristic motivation of data augmentation is to fix the size of the data set by adding *too many* all-zero encounter histories to create the data set shown in the right panel of Table 4.3 - and then analyze the augmented data set using an occupancy type model which includes both “unoccupied sites” as well as “occupied sites” at which detections did not occur. We call these $M - n$ all-zero histories “potential individuals” because they exist to be recruited (in a non-biological sense) into the population, for example during an analysis by MCMC.

To analyze the augmented data set, we recognize that it is a zero-inflated version of the known- N data set. That is, some of the augmented all-zeros are sampling zeros (corresponding to actual individuals that were missed) and some are “structural” zeros, which do not correspond to individuals in the population. For a basic closed-population model, the resulting likelihood under data augmentation - that is, for the data set of size M - is a simple zero-inflated binomial likelihood. The zero-inflated binomial model can be described “hierarchically”, by introducing a set of binary latent variables, z_1, z_2, \dots, z_M , to indicate whether each individual i is ($z_i = 1$) or is not ($z_i = 0$) a member of the population of N individuals exposed

1663 to sampling. We assume that $z_i \sim \text{Bern}(\psi)$ where ψ is the probability that an
 1664 individual in the data set of size M is a member of the sampled population - in the
 1665 sense that $1 - \psi$ is the probability of realizing a “structural zero” in the augmented
 1666 data set. The zero-inflated binomial model which arises under data augmentation
 1667 can be formally expressed by the following set of assumptions:

$$\begin{aligned} y_i | z_i = 1 &\sim \text{Bin}(K, p) \\ y_i | z_i = 0 &\sim \delta(0) \\ z_i &\stackrel{iid}{\sim} \text{Bern}(\psi) \\ \psi &\sim \text{Unif}(0, 1) \\ p &\sim \text{Unif}(0, 1) \end{aligned}$$

1668 for $i = 1, \dots, M$, where $\delta(0)$ is a point mass at $y = 0$.

1669 We note that N is no longer an explicit parameter of this model. Instead, we
 1670 estimate ψ and functions of the latent variables. In particular, under the assump-
 1671 tions of the zero-inflated model, $z_i \stackrel{iid}{\sim} \text{Bern}(\psi)$; therefore, N is a function of these
 1672 latent variables:

$$N = \sum_{i=1}^M z_i.$$

1673 Further, we note that the latent z_i parameters can be removed from the model by
 1674 integration, in which case the joint probability of the data is

$$\Pr(y_1, \dots, y_M | p, \psi) = \prod_{i=1}^M \psi \text{Bin}(y_i | K, p) + I(y_i = 0)(1 - \psi) \quad (4.2.1)$$

1675 Which can be maximized directly to obtain the MLEs of the structural parameters
 1676 ψ and p or those of other more complex models (e.g., see Royle, 2006). We could
 1677 estimate these parameters and then use them to obtain an estimator of N using
 1678 the so-called “Best unbiased predictor” (see Royle and Dorazio, 2011).

1679 4.2.2 Model M_0 in BUGS

1680 For model M_0 in which we can aggregate the encounter data to individual-specific
 1681 encounter frequencies, the augmented data are given by the vector of frequencies
 1682 $(y_1, \dots, y_n, 0, 0, \dots, 0)$. The zero-inflated model of the augmented data combines
 1683 the model of the latent variables, $z_i \sim \text{Bern}(\psi)$ with the conditional-on- z binomial
 1684 model:

$$\begin{aligned} y_i | z_i = 0 &\sim \delta(0) \\ y_i | z_i = 1 &\sim \text{Bin}(K, p) \end{aligned}$$

Table 4.3. Hypothetical occupancy data set (left), capture-recapture data in standard form (center), and capture-recapture data augmented with all-zero capture histories (right).

Occupancy data				Capture-recapture				Augmented C-R			
site	k=1	k=2	k=3	ind	k=1	k=2	k=3	ind	k=1	k=2	k=3
1	0	1	0	1	0	1	0	1	0	1	0
2	1	0	1	2	1	0	1	2	1	0	1
3	0	1	0	.	0	1	0	3	1	0	1
4	1	0	1	.	1	0	1	4	1	0	1
5	0	1	1	.	0	1	1	5	1	0	1
.	0	1	1	.	0	1	1	.	0	1	1
.	1	1	1	.	1	1	1	.	0	1	1
.	1	1	1	.	1	1	1	.	1	1	1
.	1	1	1	.	1	1	1	.	1	1	1
n	1	1	1	n	1	1	1	n	1	1	1
.	0	0	0					.	0	0	0
.	0	0	0					.	0	0	0
	0	0	0						0	0	0
	0	0	0						0	0	0
	0	0	0						0	0	0
	0	0	0					N	0	0	0
.	0	0	0					.	0	0	0
.	0	0	0						0	0	0
M	0	0	0					.	0	0	0
							
							
							
								M	0	0	0

1685 It is convenient to express the conditional-on- z observation model concisely as:

$$y_i|z_i \sim \text{Bin}(K, pz_i)$$

1686 Thus, if $z_i = 0$ then the success probability of the binomial distribution is identically
 1687 0 whereas, if $z_i = 1$, then the success probability is p . This is useful in describing
 1688 the model in the **BUGS** language, as shown below. Note the last line of the
 1689 model specification here provides the expression for computing N from the data
 1690 augmentation variables z_i .

```

1691 p ~ dunif(0,1)
1692 psi~dunif(0,1)
1693
1694 # nind = number of individuals captured at least once
1695 # nz = number of uncaptured individuals added for PX-DA
1696 for(i in 1:(nind+nz)) {
1697   z[i]~dbern(psi)
1698   mu[i]<-z[i]*p
1699   y[i]~dbin(mu[i],K)
1700 }
1701
1702 N<-sum(z[1:(nind+nz)])
```

1703 Specification of a more general model in terms of the individual encounter obser-
 1704 vations y_{ik} is not much more difficult than for the individual encounter frequencies.
 1705 We define the observation model by a double loop and change the indexing of things
 1706 accordingly, i.e.,

```

1707 for(i in 1:(nind+nz)) {
1708   z[i]~dbern(psi)
1709   for(k in 1:K){
1710     mu[i,k]<-z[i]*p
1711     y[i,k]~dbin(mu[i,k],1)
1712   }
1713 }
```

1714 In this manner, it is straightforward to incorporate covariates on p (see discussion
 1715 of this below and also chapt. 8 (REF XYZ) and consider other extensions.

1716 4.2.3 Formal development of data augmentation

1717 Use of DA for solving inference problems with unknown N can be justified as
 1718 originating from the choice of uniform prior on N . The $\text{Unif}(0, M)$ prior for N is
 1719 innocuous in the sense that the posterior associated with this prior is equal to the
 1720 likelihood for sufficiently large M . One way of inducing the $\text{Unif}(0, M)$ prior on N

1721 is by assuming the following hierarchical prior:

$$\begin{aligned} N &\sim \text{Bin}(M, \psi) \\ \psi &\sim \text{Unif}(0, 1) \end{aligned} \tag{4.2.2}$$

1722 which includes a new model parameter ψ . This parameter denotes the probability
 1723 that an individual in the super-population of size M is a member of the population
 1724 of N individuals exposed to sampling. The model assumptions, specifically the
 1725 multinomial model (eq. XYZ) and eq. 4.2.2, may be combined to yield a reparam-
 1726 eterization of the conventional model that is appropriate for the augmented data
 1727 set of known size M :

$$(n_1, n_2, \dots, n_K) \sim \text{Multin}(M, \psi\pi(1), \psi\pi(2), \dots, \psi\pi(K)) \tag{4.2.3}$$

1728 This arises by removing N from Eq. multinomial XYZ by integrating over the
 1729 binomial prior distribution for N . Thus, the models we analyze under data aug-
 1730 mentation arise formally by removing the parameter N from the ordinary model -
 1731 the model conditional on N - by integrating over a binomial prior distribution for
 1732 N .

1733 Note that the $M - n$ unobserved individuals in the augmented data set have
 1734 probability $\psi\pi(0) + (1 - \psi)$, indicating that these unobserved individuals are a
 1735 mixture of individuals that are sampling zeros ($\psi\pi_0$, and belong to the population
 1736 of size N) and others that are “structural zeros” (occurring in the augmented
 1737 data set with probability $1 - \psi$). In Eq. 4.2.3 N has been eliminated as a formal
 1738 parameter of the model by marginalization (integration) and replaced with the new
 1739 parameter ψ , which we will call the “data augmentation parameter.” However, the
 1740 full likelihood containing both N and ψ can be analyzed (see Royle et al., 2007).

1741 4.2.4 Remarks on Data Augmentation

1742 Data augmentation may seem like a strange and mysterious black-box, and likely
 1743 it is unfamiliar to most people even those with extensive experience with capture-
 1744 recapture models. However, it really is a formal reparameterization of capture-
 1745 recapture models in which N is removed from the ordinary (conditional-on- N)
 1746 model by integration. In the case of Model M0, data augmentation produces the
 1747 zero-inflated binomial which is distinct from the original observation model, but
 1748 only in the sense that it embodies, explicitly, the $\text{Unif}(0, M)$ prior for N . Choice of
 1749 M might be cause for some concern related to potential sensitivity to choice of M .
 1750 The guiding principle is that it should be chosen large enough so that the posterior
 1751 for N is not truncated, but no larger because large values entail more computational
 1752 burden. It seems likely that the properties of the Markov chains should be affected
 1753 by M and so some optimality might exist (Gopalaswamy, 2012), as in occupancy
 1754 models (Mackenzie and Royle, 2005). Formal analysis of this is required.

We emphasize the motivation for data augmentation being that it produces a data set of fixed size, so that the parameter dimension in any capture-recapture model is also fixed. As a result, MCMC is a relatively simple proposition using standard Gibbs Sampling. Consider the simplest context - analyzing Model M0 using the occupancy model. In this case, DA converts Model M0 to a basic occupancy model and the parameters p and ψ have known full-conditional distributions (in fact, beta distributions) that can be sampled from directly. Furthermore, the data augmentation variables - the latent data augmentation variables z , can be sampled from Bernoulli full conditionals. MCMC is not too much more difficult for complicated models - sometimes the hyperparameters need to be sampled using a Metropolis-Hastings step, but nothing more sophisticated than that is required.

There are other approaches to analyzing models with unknown N , using reversible jump MCMC (RJMCMC) or other so-called “trans-dimensional” (TD) algorithms² (Durbin and Elston, 2012; King, missing; Schofield and Barker, missing). What distinguishes DA from RJMCMC and related TD methods is that DA is used to create a distinctly new model that is unconditional on N and we (usually) analyze the unconditional model. The various TD/RJMCMC approaches seek to analyze the conditional-on- N model in which the dimensional of the parameter space is a variable function of N . TD/RJMCMC approaches might appear to have the advantage that one can model N explicitly or consider alternative priors for N . However, despite that N is removed as an explicit parameter in DA, it is possible to develop hierarchical models that involve structure on N (Converse and Royle, 2010; Royle et al., 2011a) which we consider in chapt. XYZ.

4.2.5 Example: Black Bear Study on Fort Drum

To illustrate the analysis of Model M0 using data augmentation, we use a data set collected at Fort Drum Military Installation in upstate New York by the Department of Defense, Cornell University and colleagues. These data have been analyzed in various forms by Gardner (2009); Gardner et al. (2010), and Wegan (missing). The specific data used here are encounter histories on 47 individuals obtained from an array of 38 baited “hair snares” (Fig. 4.1) during June and July 2006. Barbed wire traps were baited and checked for hair samples each week for eight weeks, thus we have $K = 8$ sample intervals. The data are provided on the Web Supplement and the analysis can be set up and run as follows. Here, the data were augmented with $M - n = 128$ ($M = 175$) all-zero encounter histories.

```
# Consider adding comments to your code.
## Good idea. This will be done in final draft
trapmat<-read.csv("FDtrapmat.csv")
bearArray<-source("FDbeararray.R")$value
nind<-dim(bearArray)[1]
```

²Look these citations up in Royle-Dorazio EURING paper

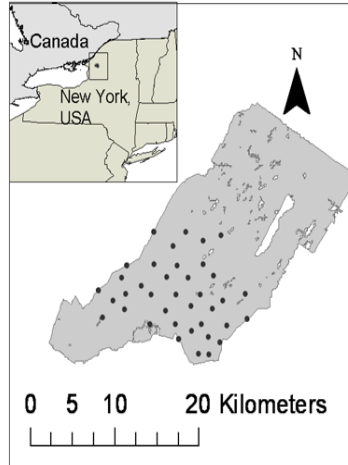


Figure 4.1. Fort Drum study area and hair snare locations.

```

1794 K<-dim(bearArray)[3]
1795 ntraps<-dim(bearArray)[2]
1796
1797 M=175
1798 nz<-M-nind
1799
1800 Xaug <- array(0, dim=c(M,ntraps,K))
1801 Xaug[1:nind,,]<-bearArray
1802 y<- apply(Xaug,c(1,3),sum)
1803 y[y>1]<-1
1804 ytot<-apply(y,1,sum) # total encounters out of K

```

Note that the raw data, \mathbf{y} , is an $M \times K$ array of individual encounter events (i.e., $y_{ik} = 1$ if individual i was encountered in any trap and 0 otherwise). For $i = 48, \dots, 175$, $y_{ik}=0$ as these are augmented observations. For Model M0 it is sufficient to reduce the data to individual encounter frequencies which we have labeled \mathbf{y}_{tot} above. The BUGS model file along with commands to fit the model are as follows:

```

1811 set.seed(2013) # to obtain the same results each time
1812 data0<-list(y=y,M=M,K=K)
1813 params0<-list('psi','p','N')
1814 zst=c(rep(1,nind),rbinom(M-nind, 1, .5))
1815 inits = function() {list(z=zst, psi=runif(1), p=runif(1)) }
1816
1817 cat("

```



```

1818 model {
1819
1820   psi~dunif(0, 1)
1821   p~dunif(0,1)
1822
1823   for (i in 1:M){
1824     z[i]~dbern(psi)
1825     for(k in 1:K){
1826       tmp[i,k]<-p*z[i]
1827       y[i,k]~dbin(tmp[i,k],1)
1828     }
1829   }
1830   N<-sum(z[1:M])
1831 }
1832 ",file="modelM0.txt")
1833
1834 fit0 = bugs(data0, inits, params0, model.file="modelM0.txt",
1835            n.chains=3, n.iter=2000, n.burnin=1000, n.thin=1,
1836            debug=TRUE,working.directory=getwd())

```

1837 The posterior summary statistics from this analysis are as follows:

```

1838 > print(fit0,digits=2)
1839 Inference for Bugs model at "modelM0.txt", fit using WinBUGS,
1840 3 chains, each with 2000 iterations (first 1000 discarded)
1841 n.sims = 3000 iterations saved
1842
1843      mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
1844 psi      0.29 0.04  0.22  0.26  0.29  0.31  0.36   1 3000
1845 p        0.30 0.03  0.25  0.28  0.30  0.32  0.35   1 3000
1846 N        49.94 1.99 47.00 48.00 50.00 51.00 54.00   1 3000
1847 deviance 489.05 11.28 471.00 480.45 488.80 495.40 513.70   1 3000
1848
1849 [.. some output deleted ..]

```

1849 **WinBUGS** did well in choosing an MCMC algorithm for this model – we have
1850 $\hat{R} = 1$ for each parameter, and an effective sample size of 3000, equal to the total
1851 number of posterior samples. We see that the posterior mean of N under this model
1852 is 49.94 and a 95% posterior interval is (48, 54). We revisit these data later in the
1853 context of more complex models.

1854 In order to obtain an estimate of density, D , we need an area to associate with
1855 the estimate of N , and commonly used procedures to conjure up such an area
1856 include buffering the trap array by the home range radius, often estimated by the
1857 mean maximum distance moved (MMDM)³, 1/2 MMDM (Dice, 1938) or directly
1858 from telemetry data (REF XXX NEED REF HERE XXXXX). Typically, the trap
1859 array is defined by the convex hull around the trap locations, and this is what we

³really MMDM? How can this be an estimate of the home range radius? Reference for this?

1860 applied a buffer to. We computed the buffer by using an estimate of the mean female
 1861 home range radius (2.19 km) estimated from telemetry studies (Bales et al., 2005)
 1862 instead of using an estimate based on our relatively more sparse recapture data⁴.
 1863 For the Fort Drum study, the convex hull has area 157.135 km^2 , and the buffered
 1864 convex hull has area 277.011 km^2 . To create this we used functions contained in the
 1865 **R** package **rgeos** and created a utility function **bcharea** which is in our **R** package
 1866 **scrbook**. The commands are as follows:

```
1867 library("rgeos")
1868
1869 bcharea<-function(buff,traplocs){
1870   p1<-Polygon(rbind(traplocs,traplocs[1,]))
1871   p2<-Polygons(list(p1=p1),ID=1)
1872   p3<-SpatialPolygons(list(p2=p2))
1873   p1ch<-gConvexHull(p3)
1874   bp1<-gBuffer(p1ch, width=buff)
1875   plot(bp1, col='gray')
1876   plot(p1ch, border='black', lwd=2, add=TRUE)
1877   gArea(bp1)
1878 }
1879
1880 bcharea(2.19,traplocs=trapmat)
```

1881 The resulting buffered convex hull is shown in Fig. 4.2.

1882 To conjure up a density estimate under model M_0 , we compute the appropriate
 1883 posterior summary of N and the prescribed area (277.011 km^2):

```
1884 > summary(fit0$sims.list$N/277.011)
1885   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1886 0.1697 0.1733 0.1805 0.1803 0.1841 0.2130
1887
1888 > quantile(fit0$sims.list$N/277.011,c(0.025,0.975))
1889      2.5%      97.5%
1890 0.1696684 0.1949381
```

1891 which yields a density estimate of about 0.18 ind/ km^2 , and a 95% Bayesian confi-
 1892 dence interval of (0.170, 0.195).

1893 The obvious limitation of this estimate and, indeed, of the whole process, is that
 1894 our choice of “area” is completely subjective - which area should we use? MMDM?
 1895 One-half MMDM? Estimated from telemetry data? And, furthermore, how certain
 1896 are we of this area? Can we quantify our uncertainty about this quantity? More
 1897 important, what exactly is the meaning of this area and, in this context, how do
 1898 we gauge bias and/or variance of “estimators” of it? (i.e., what is it estimating?).

⁴BETH: Why?

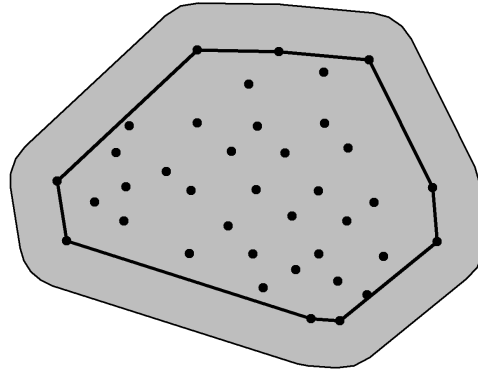


Figure 4.2. buffered convex hull of the bear hair snare array

4.3 TEMPORALLY VARYING AND BEHAVIORAL EFFECTS

The purpose of this chapter is mainly to emphasize the central importance of the binomial model in capture-recapture and so we have considered models for individual encounter frequencies - the number of times individuals are captured out of K samples. Sometimes it is not acceptable to aggregate the encounter data for each individual - such as when encounter probability varies over time among samples. A type of time-varying response that seems relevant in most capture-recapture studies is “effort” such as amount of search time, number of observers, or trap effort or when p depends on date (Kéry et al., 2010; Gardner et al., 2010). A common situation is that in which there exists a “behavioral response” to trapping (even if the animal is not physically trapped).

Behavioral response is an important concept in carnivore studies because individuals might learn to come to baited traps or avoid traps due to trauma related to being encountered. There are a number of ways to parameterize a behavioral response to encounter. The distinction between persistent and ephemeral was made by Yang and Chao (2005) who considered a general behavioral response model of the form:

$$\text{logit}(p_{ik}) = \alpha_0 + \alpha_1 * y_{i,k-1} + \alpha_2 x_{ik}$$

where x_{ik} is a covariate indicator variable of previous capture (i.e., $x_{ik} = 1$ if

captured in any previous period). Therefore, encounter probability changes depending on whether an individual was captured in the immediate previous period (ephemeral behavioral response) or in any previous period (persistent behavioral response). The former probably models a behavioral response due to individuals moving around their territory relatively slowly over time and the latter probably accommodates trap happiness due to baiting or shyness due to trauma. In spatial capture-recapture models it makes sense to consider a local behavioral response that is trap-specific (Royle et al., 2011c) - that is, the encounter probability is modified for individual traps depending on previous capture in specific traps.

Models with temporal effects are easy to describe in the **BUGS** language and analyze and we provide a number of examples in chapt. 8. XXXXX ?? XXXXX

4.4 MODELS WITH INDIVIDUAL HETEROGENEITY

Here we consider models with individual-specific encounter probability parameters, say p_i , which we model according to some probability distribution, $g(\theta)$. We denote this basic model assumption as $p_i \sim g(\theta)$. This type of model is similar in concept to extending a GLM to a GLMM but in the capture-recapture context N is unknown. The basic class of models is often referred to as “Model M_h ” but really this is a broad class of models, each being distinguished by the specific distribution assumed for p_i . There are many different varieties of Model M_h including parametric and various putatively non-parametric approaches (Burnham and Overton, 1978; Norris III and Pollock, 1996; Pledger, 2000). One important practical matter is that estimates of N can be extremely sensitive to the choice of heterogeneity model (Fienberg et al., 1999; Dorazio and Royle, 2003; Link, 2003). Indeed, Link (2003) showed that in some cases it’s possible to find models that yield precisely the same expected data, yet produce wildly different estimates of N . In that sense, N for most practical purposes is not identifiable across classes of mixture models, and this should be understood before fitting any such model. One solution to this problem is to seek to model explicit factors that contribute to heterogeneity, e.g., using individual covariate models (See 4.5 below). Indeed, spatial capture-recapture models seek to do just that, by modeling heterogeneity due to the spatial organization of individuals in relation to traps or other encounter mechanism. For additional background and applications of Model M_h see Royle and Dorazio (2008, chapt. 6) and Kery and Schaub (2011, chapt. 6).

Model M_h has important historical relevance to spatial capture-recapture situations (Karanth, 1995) because investigators recognized that the juxtaposition of individuals with the array of trap locations should yield heterogeneity in encounter probability, and thus it became common to use some version of Model M_h in spatial trapping arrays to estimate N . While this doesn’t resolve the problem of not knowing the area relevant to N , it does yield an estimator that accommodates the heterogeneity in p induced by the spatial aspect of capture-recapture studies.

To see how this juxtaposition induces heterogeneity, we have to understand

the relevance of movement in capture-recapture models. Imagine a quadrat that can be uniformly searched by a crew of biologists for some species of reptile (see Royle and Young (2008)). Figure 4.3 shows a sample quadrat searched repeatedly over a period of time. Further, suppose that species exhibits some sense of spatial fidelity in the form of a home range or territory, and individuals move about their home range (home range centroids are given by the blue dots) in some kind of random fashion. It is natural to think about it in terms of a movement process and sometimes that movement process can be modeled explicitly using hierarchical models (Royle and Young, 2008; Royle et al., 2011b). Heuristically, we imagine that each individual in the vicinity of the study area is liable to experience variable exposure to encounter due to the overlap of its home range with the sampled area - essentially the long-run proportion of times the individual is within the sample plot boundaries, say ϕ . We might model the exposure of an individual to capture by supposing that $z_i = 1$ if individual i is available to be captured (i.e., within the survey plot) during any sample, and 0 otherwise. Then, $\Pr(z_i = 1) = \phi$. In the context of spatial studies, it is natural that ϕ should depend on *where* an individual lives, i.e., it should be individual-specific ϕ_i (Chandler et al., 2011). This system describes, precisely, that of “random temporary emigration” (Kendall, 1997) where ϕ_i is the individual-specific probability of being “available” for capture.

Conceptually, SCR models aim to deal with this problem of variable exposure to sampling due to movement in the proximity of the trapping array explicitly and formally with auxiliary spatial information. If individuals are detected with probability p_0 , *conditional* on $z_i = 1$, then the marginal probability of detection of individual i is

$$p_i = p_0 \phi_i$$

so we see clearly that individual heterogeneity in encounter probability is induced as a result of the juxtaposition of individuals (i.e., their home ranges) with the sample apparatus and the movement of individuals about their home range.

We will work with a specific type of Model M_h here, that in which we extend the basic binomial observation model of Model M_0 so that

$$\text{logit}(p_i) = \mu + \eta_i$$

where

$$\eta_i \sim \text{Normal}(0, \sigma_p^2)$$

We could as well write

$$\text{logit}(p_i) \sim \text{Normal}(\mu, \sigma_p^2)$$

This “logit-normal mixture” was analyzed by Coull and Agresti (1999) and elsewhere. It is a natural extension of the basic model with constant p , as a mixed GLMM, and similar models occur throughout statistics. It is also natural to consider a beta prior distribution for p_i (Dorazio and Royle, 2003) and so-called “finite-mixture” models are also popular (Norris III and Pollock, 1996; Pledger, 2000).

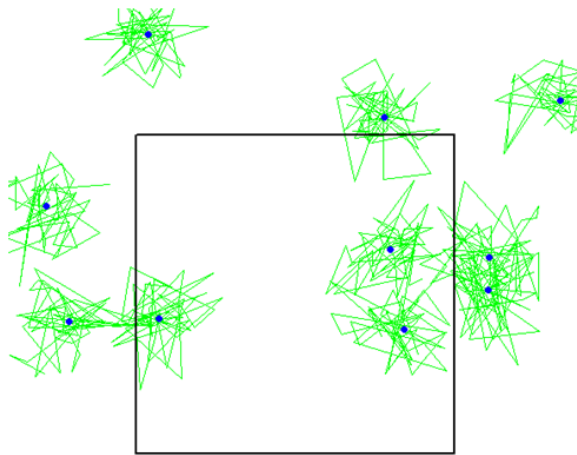


Figure 4.3. A quadrat searched for lizards and the locations of each lizard over some period of time.

1992 4.4.1 Analysis of Model Mh

1993 If N is known, it is worth taking note of the essential simplicity of Model M_h as
 1994 a binomial GLMM. This is a type of model that is widely applied in just about
 1995 every scientific discipline and using standard methods of inference based either on
 1996 integrated likelihood (Laird and Ware, 1982; Berger et al., 1999) which we discuss
 1997 in chapt. 9 or standard Bayesian methods. However, because N is not known,
 1998 inference is somewhat more challenging. We address that here using Bayesian anal-
 1999 ysis based on data augmentation (DA). Although we use data augmentation in the
 2000 context of Bayesian methods here, we note that heterogeneity models formulated
 2001 under DA are easily analyzed by conventional likelihood methods as zero-inflated
 2002 binomial mixtures (Royle, 2006) and more traditional analysis of model M_h based
 2003 on integrated likelihood, without using data augmentation, has been considered by
 2004 Coull and Agresti (1999), Dorazio and Royle (2003), and others.

2005 As with model M_0 , we have the Bernoulli model for the zero-inflation variables:
 2006 $z_i \sim \text{Bern}(\psi)$ and the model of the observations expressed conditional on the latent
 2007 variables z_i . For $z_i = 1$, we have a binomial model with individual-specific p_i :

$$y_i | z_i = 1 \sim \text{Bin}(K, p_i)$$

2008 and otherwise $y_i | z_i = 0 \sim \delta(0)$. Further, we prescribe a distribution for p_i . Here we
 2009 assume

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

2010 The basic **BUGS** description for this model, assuming a $\text{Unif}(0, 1)$ prior for $p_0 =$
 2011 $\text{logit}^{-1}(\mu)$, is given as follows:

```

2012 model{
2013
2014   p0 ~ dunif(0,1)          # prior distributions
2015   mup<- log(p0/(1-p0))
2016   taup~dgamma(.1,.1)
2017   psi~dunif(0,1)
2018
2019   for(i in 1:(nind+nz)){
2020     z[i]~dbern(psi)        # zero inflation variables
2021     lp[i] ~ dnorm(mup,taup) # individual effect
2022     logit(p[i])<-lp[i]
2023     mu[i]<-z[i]*p[i]
2024     y[i]~dbin(mu[i],J)    # observation model
2025   }
2026
2027   N<-sum(z[1:(nind+nz)]) # N is a derived parameter
2028 }
```

4.4.2 Analysis of the Fort Drum data

The logit-normal heterogeneity model was fitted to the bear data from the Fort Drum study, and we used data augmentation to produce a data set of $M = 500$ individuals. We ran the model using **JAGS** with the instructions given as follows⁵.

```
[... get data as before ....]

set.seed(2013)

cat("
model{
  p0 ~ dunif(0,1)          # prior distributions
  mup<- log(p0/(1-p0))
  taup~dgamma(.1,.1)
  sigmap<-sqrt(1/taup)
  psi~dunif(0,1)

  for(i in 1:(nind+nz)){
    z[i]~dbern(psi)        # zero inflation variables
    lp[i] ~ dnorm(mup,taup) # individual effect
    logit(p[i])<-lp[i]
    mu[i]<-z[i]*p[i]
    y[i]~dbin(mu[i],K)     # observation model
  }

  N<-sum(z[1:(nind+nz)])
}

",file="modelMh.txt")

library("rjags")
jm<- jags.model("modelMh.txt", data=data1, inits=inits, n.chains=4,
               n.adapt=1000)
jout<- coda.samples(jm, params1, n.iter=50000, thin=1)
```

CHANGE THIS TO RUN SIGMA DUNIF(0,5) PRIOR INSTEAD OF TAUU
 ANDY IS WORKING THIS SECTION RIGHT NOW. KEY ISSUE
 IS THAT BEAR DATA HAVE VERY LONG RIGHT TAIL. PSSIBLY
 NOT EVEN IDENTIFIABLE FOR LOGIT-NORMAL MODEL. NEED
 TO RUN WINBUGS AND JAGS FOR M=500 AND RUN A 200K
 RUN AND THEN MAYBE A 500K RUN TO SEE HOW THINGS
 LOOK. THEN RUN MY R CODE BELOW FOR 5 MILLION ITES
 OR SOME bs like that. I'm not sure if this is a teaching moment (Link
 2003) or if we need a different example here!

This produces the posterior distribution for N shown in Fig. 4.4. Posterior summaries of parameters are given as follows:

⁵For WinBUGS, should provide starts for lp and sigma or sometimes WinBUGS breaks


```

2072 > summary(jout)
2073 Iterations = 1001:201000
2074 Thinning interval = 1
2075 Number of chains = 4
2076 Sample size per chain = 2e+05
2077
2078 1. Empirical mean and standard deviation for each variable,
2079    plus standard error of the mean:
2080
2081      Mean      SD Naive SE Time-series SE
2082 N      108.63259 52.53176 5.873e-02      2.077726
2083 p0       0.08615  0.05919 6.618e-05      0.001950
2084 psi      0.21841  0.10615 1.187e-04      0.004141
2085 sigmap   1.94096  0.51014 5.703e-04      0.018244
2086
2087 2. Quantiles for each variable:
2088
2089      2.5%      25%      50%      75%      97.5%
2090 N      59.00000 77.00000 93.00000 121.0000 261.0000
2091 p0      0.00418  0.03852  0.07657  0.1240  0.2210
2092 psi     0.11230  0.15373  0.18920  0.2457  0.5241
2093 sigmap  1.11906  1.57643  1.87752  2.2386  3.1166

```

2094 We used $M = 500$ for this analysis and we note that while the posterior mass
2095 of N is concentrated away from this upper bound (Fig. 4.4), the posterior has an
2096 extremely long right tail, with some posterior values at the upper bound $N = 500$.
2097 Maybe or maybe not sufficient data augmentation. The model runs effectively in
2098 **WinBUGS** but sometimes with apparently inefficient mixing for reasons that may
2099 be related to bad starting values. In some cases this was resolved if we supplied
2100 starting values for the $\text{logit}(p_i)$ parameters and τ .

2101 **to do:** insert final results. longer run. more data augmentation. compare
2102 with winbugs.

2103 The posterior mode compares well with the MLE which we obtained using the **R**
2104 code contained in Panel 6.1 of Royle and Dorazio (2008). The MLE of $\log(n_0)$, the
2105 logarithm of the number of uncaptured individuals, is $\log(n_0) = 3.86$ and therefore
2106 the MLE is $\hat{N} = \exp(3.86) + 47 = 94.47$ consistent with the apparent mode in
2107 Fig. 4.4.⁶ To convert this to density we use the buffered area as computed above
2108 (255.3 km^2) ⁷ and perform the required summary analysis on the posterior samples
2109 of N , which results in about 0.37 individuals/ km^2 . The reader should carry out
2110 this analysis to confirm the estimates, and also obtain the 95% confidence interval.

⁶We note that the result is inconsistent with Gardner et al. (2009) who reported an MLE of 104.1 ($\text{density} = 0.437 \text{ inds}/\text{km}^2$) although we do not know the reason for this at the present time.

⁷WRONG #

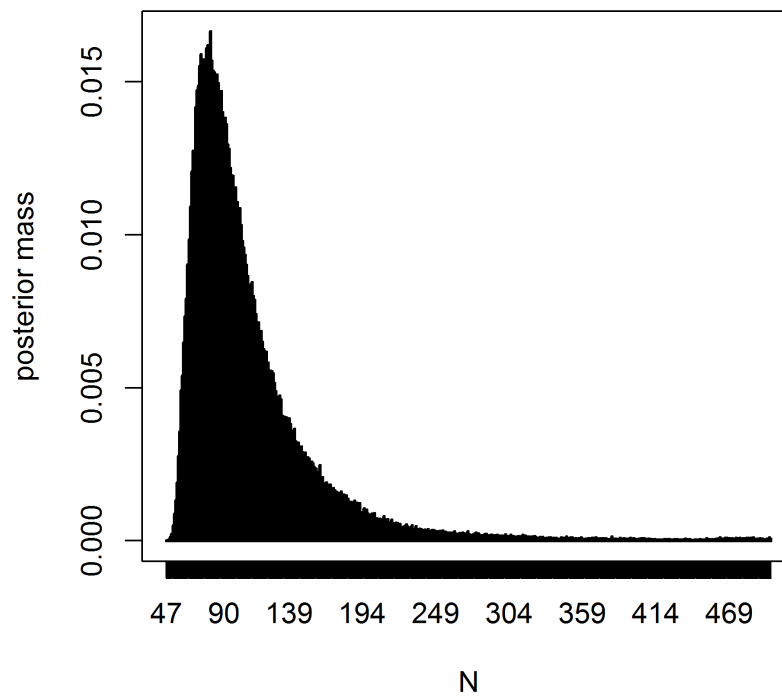


Figure 4.4. Posterior of N for Fort Drum bear study data under the logit-normal version of model M_h . From WinBUGS output. 200k samples.

4.4.3 Building your own MCMC algorithm

For fun, we construct our own MCMC algorithm using a Metropolized Gibbs sampler for Model M_h . In chapter 10 we devise MCMC algorithms for spatial capture-recapture models and the basic conceptual and technical considerations are entirely analogous to Model M_h .

To begin, we first collect all of our model components which are as follows: $[y_i|p_i, z_i]$, $[p_i|\mu_p, \sigma_p]$, and $[z_i|\psi]$ for *each* $i = 1, 2, \dots, M$ and then prior distributions $[\mu_p]$, $[\sigma_p]$ and $[\psi]$. The joint posterior distribution of all unknown quantities in the model is proportional to the joint distribution of all elements y_i, p_i, z_i and also the prior distributions of the prior parameters:

$$\left\{ \prod_{i=1}^M [y_i|p_i, z_i][p_i|\mu_p, \sigma_p][z_i|\psi] \right\} [\mu_p, \sigma_p, \psi]$$

For prior distributions, we assume that μ_p, σ_p, ψ are mutually independent and for μ_p and σ_p we use improper uniform priors, and $\psi \sim \text{Unif}(0, 1)$. Note that the likelihood contribution for each individual, when conditioned on p_i and z_i , does not depend on ψ, μ_p , or σ_p . As such, the full-conditionals for the structural parameters ψ only depends on the collection of data augmentation variables z_i , and that for μ_p and σ_p will only depends on the collection of latent variables $p_i; i = 1, 2, \dots, M$. The full conditionals for all the unknowns are as follows:

(1) For p_i :

$$[p_i|y_i, \mu_p, \sigma_p, z_i = 1] \propto [y_i|p_i][p_i|\mu_p, \sigma_p^2] \text{ if } z_i = 1 \\ [p_i|\mu_p, \sigma_p] \text{ if } z_i = 0$$

(2) for z_i :

$$z_i|\cdot \propto [y_i|z_i * p_i] \text{Bern}(z_i|\psi)$$

(3) For μ_p :

$$[\mu_p|\cdot] \sim \prod_i [p_i|\cdot] * \text{const}$$

(4) For σ_p :

$$[\sigma_p|\cdot] \sim \prod_i [p_i|\cdot] * \text{const}$$

(5) For ψ :

$$\psi|\cdot \sim \text{Beta}(1 + \sum z_i, 1 + M - \sum z_i)$$

What we've done here is identify each of the full conditional distributions in sufficient detail to toss them into our Metropolis-Hastings algorithm. With the exception of ψ which has a convenient analytic solution – it is a beta distribution which we can easily sample directly. In truth, we could also sample μ_p and σ_p^2 directly with

certain choices of prior distributions. For example, if $\mu_p \sim \text{Normal}(0, 1000)$ then the full conditional for μ_p is also normal, etc.. We implement an MCMC algorithm for this model in the following block of **R** code. The basic structure is: initialize the parameters and create any required output or intermediate “holders”, and then begin the main MCMC loop which, in this case, generates 100000 samples.

```

2142
2143 ## obtain the bear data by executing the previous data grabbing
2144 ## function
2145
2146 temp<-getdata()
2147 M<-temp$M
2148 K<-temp$K
2149 ytot<-temp$ytot
2150
2151
2152 ###
2153 ### MCMC algorithm for Model Mh
2154
2155 out<-matrix(NA,nrow=100000,ncol=4)
2156 dimnames(out)<-list(NULL,c("mu","sigma","psi","N"))
2157 lp<- rnorm(M,-1,1)
2158 p<-expit(lp)
2159 mu<- -1
2160 p0<-exp(mu)/(1+exp(mu))
2161 sigma<- 1
2162 psi<- .5
2163 z<-rbinom(M,1,psi)
2164 z[ytot>0]<-1
2165
2166 for(i in 1:100000){
2167
2168   ### update the logit(p) parameters
2169   lpc<- rnorm(M,lp,1) # 0.5 is a tuning parameter
2170   pc<-expit(lpc)
2171   lik.curr<-log(dbinom(ytot,K,z*p)*dnorm(lp,mu,sigma))
2172   lik.cand<-log(dbinom(ytot,K,z*pc)*dnorm(lpc,mu,sigma))
2173   kp<- runif(M) < exp(lik.cand-lik.curr)
2174   p[kp]<-pc[kp]
2175   lp[kp]<-lpc[kp]
2176
2177   p0c<- rnorm(1,p0,.05)
2178   if(p0c>0 & p0c<1){

```

```

2179 muc<-log(p0c/(1-p0c))
2180 lik.curr<-sum(dnorm(lp,mu,sigma,log=TRUE))
2181 lik.cand<-sum(dnorm(lp,muc,sigma,log=TRUE))
2182 if(runif(1)<exp(lik.cand-lik.curr)) {
2183   mu<-muc
2184   p0<-p0c
2185 }
2186 }
2187
2188 sigmac<-rnorm(1,sigma,.5)
2189 if(sigmac>0){
2190   lik.curr<-sum(dnorm(lp,mu,sigma,log=TRUE))
2191   lik.cand<-sum(dnorm(lp,mu,sigmac,log=TRUE))
2192   if(runif(1)<exp(lik.cand-lik.curr))
2193     sigma<-sigmac
2194 }
2195
2196 ### update the z[i] variables
2197 zc<- ifelse(z==1,0,1) # candidate is 0 if current = 1, etc..
2198 lik.curr<- dbinom(ytot,K,z*p)*dbinom(z,1,psi)
2199 lik.cand<- dbinom(ytot,K,zc*p)*dbinom(zc,1,psi)
2200 kp<- runif(M) < (lik.cand/lik.curr)
2201 z[kp]<- zc[kp]
2202
2203 psi<-rbeta(1, sum(z) + 1, M-sum(z) + 1)
2204
2205 out[i,]<- c(mu,sigma,psi,sum(z))
2206 }

```

2207 **Remarks:** (1) for parameters with bounded support, i.e., σ_p and p_0 , we are using a random walk candidate generator but rejecting draws outside of the parameter space. (2) We mostly use Metropolis-Hastings except for the data augmentation parameter ψ which we sample directly from its full-conditional distribution which is a beta distribution. (3) Even the latent data augmentation variables z_i are updated using Metropolis-Hastings although they too can be updated directly from their full-conditional.

2214 4.4.4 Exercises related to model Mh

- 2215 (1) Enclose the MCMC algorithm in an R function and provide arguments for some of the parameters of the function that a user might wish to modify.
- 2216
- 2217 (2) Execute the function and compare the results to those generated from WinBUGS in the previous section
- 2218

- 2219 (3) Note that the prior distribution for the “mean” parameter is given on $p_0 =$
 2220 $\exp(\mu)/(1 + \exp(\mu))$. Reformulate the algorithm with a flat prior on μ and see
 2221 what happens. Contemplate this.
 2222 (4) Using Bayes rule, figure out the full conditional for z_i so that you don’t have
 2223 to use MH for that one. It might be more efficient. Is it?

4.5 INDIVIDUAL COVARIATE MODELS: TOWARD SPATIAL CAPTURE-RECAPTURE

2224 A standard situation in capture-recapture models is when an individual covariate
 2225 is measured, and this covariate is thought to influence encounter probability. As
 2226 with other closed population models, we begin with the basic binomial observation
 2227 model:

$$y_i \sim \text{Bin}(K, p_i)$$

2228 and we assume also a model for encounter probability according to:

$$\text{logit}(p_i) = \alpha_0 + \alpha_1 x_i$$

2229 Classical examples of covariates influencing detection probability are type of animal
 2230 (juvenile/adult or male/female), a continuous covariate such as body mass (Royle
 2231 and Dorazio, 2008, chapt. 6), or a discrete covariate such as group or cluster
 2232 size. For example, in models of aerial survey data, it is natural to model detection
 2233 probabilities as a function of the observation-level individual covariate, “group size”
 2234 (Royle, 2008, 2009; Langtimm, 2010).

2235 Such “individual covariate models” are similar in structure to Model M_h , except
 2236 that the individual effects are *observed* for the n individuals that appear in the
 2237 sample. These models are important here because spatial capture-recapture models
 2238 are precisely a form of individual covariate model, an idea that we will develop
 2239 here and elsewhere. Specifically, they are such models, but where the individual
 2240 covariate is a partially observed latent variable similar.. That is, unlike Model M_h ,
 2241 we do have some direct information about the latent variable, which comes from
 2242 the spatial locations/distribution of individual recaptures. More on that later.

2243 Traditionally, estimation of N in individual covariate models is achieved using
 2244 methods based on ideas of unequal probability sampling (i.e., Horwitz-Thompson
 2245 estimation), see Huggins (1989) and Alho (1990). An estimator of N is

$$\hat{N} = \sum_i \frac{1}{\tilde{p}_i}$$

2246 where \tilde{p}_i is the probability that individual i appeared in the sample. That is,
 2247 $\tilde{p}_i = \Pr(y_i > 0)$. In practice, \tilde{p}_i is estimated from the conditional-likelihood formed
 2248 by the encounter histories. Namely,

$$\Pr(y_i | y_i > 0) = \Pr(y_i) / \Pr(y_i > 0)$$

2249 where we substitute

$$\Pr(y_i > 0) = (1 - (1 - p_i)^K)$$

2250 with

$$\text{logit}(p_i) = \alpha_0 + \alpha_1 x_i$$

2251 Here we take a formal model-based approach to Bayesian analysis of such models
 2252 using data augmentation (Royle, 2009). Classical likelihood analysis of the so-
 2253 called “full likelihood” is covered in some detail by Borchers et al. (2002). For
 2254 Bayesian analysis of individual covariate models, because the individual covariate
 2255 is unobserved for the $N - n$ uncaptured individuals, we require a model to describe
 2256 variation among individuals, essentially allowing the sample to be extrapolated to
 2257 the population. For our present purposes, we consider a continuous covariate and
 2258 we assume that it has a normal distribution:

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

2259 Data augmentation can be applied directly to this class of models. In particular,
 2260 reformulation of the model under DA yields a basic zero-inflated binomial model of
 2261 the form:

$$\begin{aligned} z_i &\sim \text{Bern}(\psi) \\ y_i | z_i = 1 &\sim \text{Bin}(K, p_i) \\ y_i | z_i = 0 &\sim \delta(0) \end{aligned}$$

2262 In addition, we assume that p_i is functionally related to a covariate x_i , e.g., by the
 2263 logit model given above, and we assume a distribution for x_i appropriate for the
 2264 context.

2265 Fully spatial capture-recapture models essentially use this formulation with a
 2266 latent covariate that is directly related to the individual detection probability (see
 2267 next Section). As with the previous models, implementation is trivial in the BUGS
 2268 language. The BUGS specification is very similar to that for model M_h , but we
 2269 require the distribution of the covariate to be specified, along with priors for the
 2270 parameters of that distribution.

2271 4.5.1 Example: Location of capture as a covariate.

2272 If we had a regular grid of traps over some closed geographic system then we imagine
 2273 that the average location of capture would be a decent estimate (heuristically) of
 2274 an individual’s home range center. Intuitively some measure of typical distance
 2275 from home range center to traps for an individual should be a decent covariate to
 2276 explain heterogeneity in encounter probability, i.e., individuals with more exposure
 2277 to traps should have higher encounter probabilities and vice versa. A version of

this idea was put forth by Boulanger and McLellan (2001) (see also Ivan (2012)), but using the Huggins-Alho estimator and with covariate “distance to edge” of the trapping array. A limitation of this basic approach is that it does not provide a solution to the problem that the trap area is fundamentally ill-defined, nor does it readily accommodate the inherent and heterogeneous variation in this measured covariate. Here, we provide an example of this type of heuristically motivated approach using the fully model-based individual covariate model described above analyzed by data augmentation. We take a slightly different approach than that adopted by Boulanger and McLellan (2001). By analyzing the full likelihood and placing a prior distribution on the individual covariate, we resolve the problem of having an ill-defined area over which the population size is distributed. After you read later chapters of this book, it will be apparent that SCR models represent a formalization of this heuristic procedure.

For our purposes here, we define $x_i = ||s_i - x_0||$ where s_i is the average encounter location of individual i and x_0 is the centroid of the trap array. Conceptually, individuals in the middle of the array should have higher probability of encounter and, as x_i increases, p_i should therefore decrease. We note that we have defined s_i in terms of a sample quantity - the observed mean - which is ad hoc but maybe satisfactory under the circumstances. That said, for an expansive, dense trapping grid then we might expect the sample mean encounter location to be a good estimate of home range center but, clearly this is biased for individuals that live around the edge (or off) the trapping array. Regardless, it should be good enough for our present purposes of demonstrating this heuristically appealing application of an individual covariate model. A key point is that s_i is missing for each individual that is not encountered and thus so is x_i . Thus, it is a latent variable, or random effect, and we need therefore to specify a probability distribution for it. As a measurement of distance we know it must be positive-valued. Suppose further than we imagine no individual could have a home range radius larger than D_{max} . As such, we think a reasonable distribution for this individual covariate is

$$x_i \sim \text{uniform}(0, D_{max})$$

where D_{max} is a specified constant. In practice, people have used distance from edge of the trap array but that is less easy to define and compute.

Fort Drum Bear Study

We have to do a little bit of data processing to fit this individual covariate model to the Fort Drum data. To compute the average location of capture for each individual and the distance from the centroid of the trap array, we execute the following R instructions:

```
avg.s<-matrix(NA,nrow=nind,ncol=2)
for(i in 1:nind){
  tmp<-NULL
```



```

2317 for(j in 1:T){
2318   aa<-bearArray[i,,j]
2319   if(sum(aa)>0){
2320     aa<- trapmat[aa>0,]
2321     tmp<-rbind(tmp,aa)
2322   }
2323 }
2324 avg.s[i,]<-c(mean(tmp[,1]),mean(tmp[,2]))
2325 }
2326 Cx<-mean(trapmat[,1])
2327 Cy<-mean(trapmat[,2])
2328 avg.s<-rbind(avg.s,matrix(NA,nrow=nz,ncol=2))
2329 xcent<- sqrt( (avg.s[,1]-Cx)^2 + (avg.s[,2]-Cy)^2)

```

2330 To define the maximum distance (maxD) from the centroid, we use that of the
 2331 farthest trap, and so maxD is computed as follows:

```

2332 minx<- min(trapmat[,1]-Cx)
2333 maxx<-max(trapmat[,1]-Cx)
2334 miny<- min(trapmat[,2]-Cy)
2335 maxy<- max(trapmat[,2]-Cy)
2336 # most extreme point determines maxD
2337 ul<- c(minx,maxy)
2338 maxD<- sqrt( (ul[1]-0)^2 + (ul[2]-0)^2)

```

2339 For the bear data the maxD was about 11.5 km. As such, the model described
 2340 above will produce an estimate of the population size of bears within 11.5 units of
 2341 the trap centroid⁸. The BUGS model specification and R commands to package
 2342 the data and fit the model are as follows:

```

2343 cat("
2344 model{
2345   p0 ~ dunif(0,1)          # prior distributions
2346   mup<- log(p0/(1-p0))
2347   psi~dunif(0,1)
2348   beta~dnorm(0,.01)
2349
2350   for(i in 1:(nind+nz)){
2351     xcent[i]~dunif(0,maxD)
2352     z[i]~dbern(psi)        # DA variables
2353     lp[i] <- mup + beta*xcent[i] # individual effect
2354     logit(p[i])<-lp[i]

```

⁸To be convincing this might need a little bit of hand-holding

```

2355     mu[i]<-z[i]*p[i]
2356     y[i]~dbin(mu[i],K) # observation model
2357 }
2358 N<-sum(z[1:(nind+nz)])
2359 }
2360 ",file="modelMcov.txt")
2361 data2<-list(y=ytot,nz=nz,nind=nind,K=T,xcent=xcent,maxD=11.5)
2362 params2<-list('p0','psi','N','beta')
2363 inits = function() {list(z=z, psi=psi, p0=runif(1),beta=rnorm(1) ) }
2364 fit2 = bugs(data2, inits, params2, model.file="modelMcov.txt",working.directory=getwd(),
2365             debug=T, n.chains=3, n.iter=4000, n.burnin=1000, n.thin=4)

```

Posterior summaries are given in Table ?? XYZ, and the posterior distribution of N is given in Figure XYZ. It might be perplexing that the estimated N is much lower than obtained by model Mh but there is a good explanation for this, discussed subsequently. That issue notwithstanding, it is worth pondering how this model could be an improvement (conceptually or technically) over some other model/estimator including M0 and Mh considered previously. Well, for one, we have accounted formally for heterogeneity due to spatial location of individuals relative to exposure to the trap array, characterized by the centroid of the array. Moreover, we have done so using a model that is based on an explicit mechanism, as opposed to a phenomenological one such as Model Mh. Moreover, importantly, using our new model, *the estimated N applies to an explicit area which is defined by our prescribed value of $maxD$* . That is, this area is a fixed component of the model and the parameter N therefore has explicit spatial context, as the number of individuals with home range centers less than $maxD$ from the centroid of the trap array. As such, the implied “effective trap area”⁹ for any $maxD$ is that of a circle with radius $maxD$.

```

2382 %% Not sure whether this should be a table or verbatim print-out
2383 \begin{table}
2384 \tabular{cccccccc}
2385 Node statistics
2386 node mean sd MC error 2.5% median 97.5% start sample
2387 N 58.89 5.483 0.2199 50.0 58.0 71.0 251 2250
2388 beta -0.246 0.06087 0.003892 -0.3592 -0.2457 -0.126 251 2250
2389 deviance 459.4 13.29 0.4496 435.7 458.4 487.8 251 2250
2390 p0 0.5409 0.06817 0.004052 0.4072 0.544 0.6678 251 2250
2391 psi 0.1706 0.02572 7.759E-4 0.1247 0.1692 0.2242 251 2250
2392 \end{tabular}
2393 \caption{..... xyz .....}
2394 \end{table}

```

⁹This is a bad use of this term. We have never defined ETA or ESA. What is it, exactly?

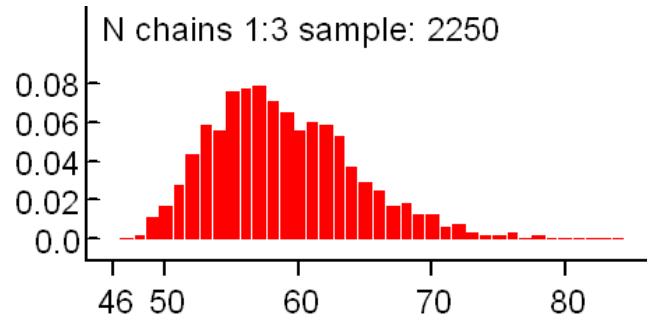


Figure 4.5. Needs a caption

2395 `\label{tab.maxD}`

2396 We'll remake this figure in R. For now, insert it as is.

2397 4.5.2 Extension of the Model

2398 One important issue in understanding the meaning of estimates produced under
 2399 the individual covariate model is that the uniform distribution on maxD implies
 2400 that density is *not constant* over space. In particular, this model implies that it
 2401 *decreases* as we move away from the centroid of the trap array. This is one reason
 2402 we have a lower estimate of density than that obtained previously and also why,
 2403 if we were to increase maxD , we would see density continue to decrease: $x[i] \sim$
 2404 $\text{Uniform}(0, \text{maxD})$ implies constant N in each distance band from the centroid but
 2405 obviously the *area* of each distance band is increasing. The reader can verify this
 2406 as a homework exercise. Obviously, the use of an individual covariate model is *not*
 2407 restricted to use of this specific distribution for the individual covariate. Clearly, it is
 2408 a bad choice and, therefore, we should think about whether we can choose a better
 2409 distribution for maxD - one that doesn't imply a decreasing density as distance
 2410 from the centroid increases. Conceptually, what we want to do is impose a prior on
 2411 distance from the centroid, x , such that density is proportional to the amount of
 2412 area in each successive distance band as you move farther away from the centroid.
 2413 In fact, there is theory that exists which tells us what the correct distribution
 2414 of x is $2x/\text{maxD}^2$. This can be derived by noting that $F(x) = \text{Pr}(X < x) =$
 2415 $\pi * x * x / \pi * \text{maxD} * \text{maxD}$. Then, $f(x) = dF/dx = 2 * x / (\text{maxD} * \text{maxD})$. This
 2416 might be called a triangular distribution, I think, which makes sense because the
 2417 incremental area in each additional distance band increases linearly with radius (i.e.,
 2418 distance from centroid). It is sometimes comforting to verify things empirically:

2419 `> u<-runif(10000,-1,1)`

```

2420 > v<-runif(10000,-1,1)
2421 > d<- sqrt(u*u+v*v)
2422 > hist(d[d<1])
2423 > hist(d[d<1],100)
2424 > hist(d[d<1],100,probability=TRUE)
2425 > abline(0,2)

```

It would be useful if we could describe this distribution in *BUGS but there is not a built-in way to do this. One possibility is to use a discrete version of the pdf. We might also be able to use what is referred to in WinBUGS jargon as the “zeros trick” (see Advanced BUGS tricks) although we haven’t pursued this approach. Instead, we consider using a discrete version and break D_{\max} into L distance classes of width δ , with probabilities proportional to $2 * x$. In particular, if the cut-points are $xg[1] = 0, xg[2], \dots, xg[L + 1] = D_{\max}$ and the interval midpoints are $xm[i] = xg[i + 1] - \delta$. Then, the interval probabilities are $p[i] = 2 * xm[i] * \delta / (D_{\max} * D_{\max})$, which we can compute once and then send them to WinBUGS as data.

The R script is as follows. In the model description the variable x (observed home range center) has been rounded so that the discrete version of the $f(x)$ can be used as described previously. The new variable labeled `xround` is actually then the integer category label in units of δ from 0. Thus, to convert back to distance in the expression for $lp[i]$, `xround[i]` has to be multiplied by δ .

```

2440 delta<-.2
2441 xround<-xcent%%delta + 1
2442 Dgrid<- seq(delta,maxD,delta)
2443 xprobs<- delta*(2*Dgrid/(maxD*maxD))
2444 xprobs<-xprobs/sum(xprobs)
2445
2446 cat("
2447 model{
2448   p0 ~ dunif(0,1)          # prior distributions
2449   mup<- log(p0/(1-p0))
2450   psi~dunif(0,1)
2451   beta~dnorm(0,.01)
2452
2453   for(i in 1:(nind+nz)){
2454     xround[i]~dcat(xprobs[])
2455     z[i]~dbern(psi)          # zero inflation variables
2456     lp[i] <- mup + beta*xround[i]*delta # individual effect
2457     logit(p[i])<-lp[i]
2458     mu[i]<-z[i]*p[i]
2459     y[i]~dbin(mu[i],K)      # observation model
2460   }
2461

```

Table 4.4. Table: Analysis of Fort Drum bear hair snare data using the individual covariate model, for different values of D_{\max} , the upper limit of the uniform distribution of ‘distance from centroid of the trap array’

D_{\max}	mn	SD
[1,] 12	0.230	0.038
[2,] 15	0.244	0.041
[3,] 17	0.249	0.044
[4,] 18	0.249	0.043
[5,] 19	0.250	0.043
[6,] 20	0.250	0.044

```

2462 N<-sum(z[1:(nind+nz)])
2463 }
2464 ",file="modelMcov.txt")

```

2465 To fit the model we do this - keeping in mind that the data objects required
 2466 below have been defined in previous analyses of this chapter:

```

2467 data2<-list(y=ytot,nz=nz,nind=nind,K=T,xround=xround,xprobs=xprobs,delta=delta)
2468 params2<-list('p0','psi','N','beta')
2469 inits = function() {list(z=z, psi=psi, p0=runif(1),beta=rnorm(1) ) }
2470 fit = bugs(data2, inits, params2, model.file="modelMcov.txt",working.directory=getwd(),
2471           debug=FALSE, n.chains=3, n.iter=11000, n.burnin=1000, n.thin=2)

```

2472 This is a useful model because it induces a clear definition of area in which
 2473 the population of N individuals reside. Under this model, that area is defined by
 2474 specification of $\max D$. We can apply the model for different values of $\max D$ and
 2475 observe that the estimated N varies with $\max D$. Fortunately, we see empirically,
 2476 that while N seems highly sensitive to the prescribed value of $\max D$, density seems
 2477 to be invariant to $\max D$ as long as it is chosen to be sufficiently large. We fit the
 2478 model for $\max D = 12$ (points in close proximity to the trap array) to 20 for and the
 2479 results are given in Table ??.

2480 We see that the posterior mean and SD of density (individuals per square km)
 2481 appear insensitive to choice of $\max D$ once we get a slight ways away from the
 2482 maximum observed value of about 11.5. The estimated density of 0.250 per km² is
 2483 actually quite a bit lower than we reported using model Mh (0.37, see section XYZ
 2484 above) for which sample area is not an explicit feature of the model. On the other
 2485 hand it is higher than that reported from Model M0 using the buffered area (0.195).
 2486 There is no basis really for comparing or contrasting these various estimates and
 2487 it would be a useful philosophical exercise for the reader to discuss this matter.
 2488 In particular, application of model M0 and Mh are distinctly *not* spatially explicit
 2489 models – the area within which the population¹⁰ resides is not defined under either
 2490 model. There is therefore no reason at all to think that the estimates produced
 2491 under either model, using a buffered area, are justifiable based on any theory. In
 2492 fact, we would get exactly the same estimate of N no matter what we declare the
 2493 area to be. On the other hand, the individual covariate model explicitly describes

¹⁰We need to look back at Chapter 1 and make sure we quit calling this “sample area” - it really isn’t that at all, but rather the area within which N resides.

a distribution for “distance from centroid” that is a reasonable and standard null model - it posits, in the absence of direct information, that individual home range centers are randomly distributed in space and that probability of detection depends on the distance between home range center and the centroid of the trap array. Under this definition of the system, we see that density is invariant to the choice of sample area which seems like a desirable feature. The individual covariate model is not ideal, however, because it does not make full use of the spatial information in the data set, i.e., the trap locations and the locations of each individual encounter.

4.5.3 Invariance of density to maxD

Under the model above, and also under models that we consider in later chapters, a general property of the estimators is that while N increases with the prescribed trap area (equivalent to $\max D$ in this case), we expect that density estimators should be invariant to this area. In the model used above, we note that $Area(\max D) = \pi * \max D * \max D$ and $E[N(\max D)] = \lambda * A(\max D)$ and thus $E[Density(\max D)] = \lambda$ which is constant. This should be interpreted as the *prior* density. Absent data, then realizations under the model will have density λ regardless of what $\max D$ is prescribed to be. As we verified empirically above, the posterior density is also invariant if $\max D$ as long as the implied area (implied by $\max D$) is large enough so that the data no longer provide information about density (i.e., “far away”), then our estimator of density should become insensitive.

4.5.4 Toward Fully Spatial Capture-recapture Models

We developed this model for the average observed location and equated it to home range center s_i . Intuitively, taking the average encounter location as an estimate of home range center makes sense but more so when the trapping grid is dense and expansive relative to typical home range sizes. However, our approach also ignored the variable precision with which each $s[i]$ is estimated and also, as noted previously, estimates of $s[i]$ around the “edge” (however we define that) are biased because the observations are truncated (we can only observe locations within the trap array). In the next Chapter we provide a further extension of this individual covariate model that definitively resolves the ad hoc nature of the individual covariate approach we took here. In that model we build a model in which $s[i]$ are regarded as latent variables and the observation locations (i.e., trap specific encounters) are linked to those latent variables with an explicit model. We note that the model fitted previously could be adapted easily to deal with s_i as a latent variable, simply by adding a prior distribution for s_i . The reader should contemplate how to do this in WinBUGS.

4.6 DISTANCE SAMPLING: A PRIMITIVE SPATIAL CAPTURE-RECAPTURE MODEL

Distance sampling is one of the most popular methods for estimating animal abundance. One of the great benefits of distance sampling is that it provides explicit estimates of *density*. The distance sampling model is a special case of a closed population model with a covariate. The covariate in this case, x_i , is the distance between an individual's location " u " and the observation location or transect. In fact, the model underlying distance sampling is precisely the same model as that which applies to the individual-covariate models, except that observations are made at only $K = 1$ sampling occasion. In a sense, distance sampling is a spatial capture-recapture model, but without the "recapture." This first and most basic spatial capture-recapture model has been used routinely for decades and, formally, it is a spatially-explicit model in the sense that it describes, explicitly, the spatial organization of individual locations (although this is not always stated explicitly) and, as a result, somewhat general models of how individuals are distributed in space can be specified (Royle, 2004b; Johnson, 2010; Sillett, 2011). As before, the distance sampling model, under data augmentation, includes a set of M zero-inflation variables z_i and the binomial model expressed conditional on z (binomial for $z = 1$, and fixed zeros for $z = 0$). In distance sampling we pay for having only a single sample (i.e., $K = 1$) by requiring constraints on the model of detection probability. A standard model is

$$\log(p_i) = b * x_i^2$$

for $b < 0$, where x_i denotes the distance at which the i th individual is detected relative to some reference location where perfect detectability ($p = 1$) is assumed. This function corresponds to the "half-normal" detection function (i.e., with $b = 1/\sigma^2$). If $K > 1$ then the intercept alpha is identifiable and such models are usually called "capture-recapture distance sampling" (Borchers, missing) and others XYZ???)

As with previous examples, we require a distribution for the individual covariate x_i . The customary choice is

$$x_i \sim \text{Uniform}(0, B)$$

wherein $B > 0$ is a known constant, being the upper limit of data recording by the observer (i.e., the point count radius, or transect half-width). In practice, this is sometimes asserted to be infinity, but in such cases the distance data are usually truncated. Specification of this distance sampling model in the BUGS language is shown in Panel 4.1. Royle and Dorazio (2008), p. xyz) provide a distance sampling example analyzed by DA using the famous Impala data.

As with the individual covariate model in the previous section, the distance sampling model can be equivalently specified by putting a prior distribution on individual *location* instead of distance between individual and observation point

```

b~dunif(0,10)
psi~dunif(0,1)

for(i in 1:(nind+nz)){
  z[i]~dbern(psi)      # DA Variables
  x[i]~dunif(0,B)      # B=strip width
  p[i]<-exp(logp[i])    # DETECTION MODEL
  logp[i]<- -((x[i]*x[i])*b)
  mu[i]<-z[i]*p[i]
  y[i]~dbern(mu[i])    # OBSERVATION MODEL
}
N<-sum(z[1:(nind+nz)])
D<- N/striparea # area of transects

```

Panel 4.1: Distance sampling model in WinBUGS, using a “half-normal” detection function.

2566 (or transect). Thus we can write the general distance sampling model as

$$\text{logit}(p[i]) = \alpha + \beta * ||u[i] - x_0||$$

2567 Along with

$$\mathbf{u}_i \sim \text{Uniform}(\mathcal{S})$$

2568 where x_0 is a fixed point (or line) and $u[i]$ is the individual’s location which is
 2569 observable for n individuals. In practice it is easier to record distance instead
 2570 of location. Basic math can be used to argue that if individuals have a uniform
 2571 distribution in space, then the distribution of Euclidean distance is also uniform.
 2572 In particular, if a transect of length L is used and x is distance to the transect then
 2573 $F(x) = \Pr(X \leq x) = L * x / L * B = x/B$ and $f(x) = dF/dx = (1/B)$. For
 2574 measurements of radial distance, see the previous section.

2575 In the context of our general characterization of SCR models (chapter 1.XYZ),
 2576 we suggested that every SCR model can be described, conceptually, by a hierarchical
 2577 model of the form:

$$[y|u][u|s][s].$$

2578 Distance sampling ignores s , and treats u as observed data¹¹. Thus, we are left
 2579 with

$$[y|u][u].$$

2580 In contrast, as we will see in the next chapters, basic SCR models (chapter 4) ignore
 2581 u and condition on s , which is not observed:

$$[y|s][s]$$

¹¹Formally we could also say that $[u] = \int [y|s][s]ds$

Since $[u]$ and $[s]$ are both assumed to be uniformly distributed, these are structurally equivalent models! The main differences have to do with interpretation of model components and whether or not the latent variables are observable (in distance sampling they are).

So why bother with SCR models when distance sampling yields density estimates and accounts for spatial heterogeneity in detection? For one, imagine try to collect distance sampling data on tigers! Clearly, distance sampling requires that one can collect large quantities of distance data, which is not always possible. For tigers, it is much easier, efficient, and safer to employ camera traps or tracking plates and then apply SCR models. Furthermore, as we will see in Ch XYZ, SCR models can use distance data to estimate all the parameters of our enchilada, allowing us to study distribution, movement, and density. Thus, SCR models are much more flexible than distance sampling models, and can accommodate data from virtually all animal survey designs.

4.6.1 Example: Muntjac deer survey from Nagarahole, India

Here we fit distance sampling models to distance sampling data on the muntjac deer (*Muntiacus muntjak*) collected in the year 2004 from Nagarahole National Park in southern India (Kumar, missing)(Kumar et al. unpublished data). The muntjac is a solitary species and distance measurements were made on 57 groups that were largely singletons with XYZ pairs of individuals. Commands for reading in and organizing the data for WinBUGS, followed by writing the model to a text file. Note that the total sampled area of the transects is fed in as “striparea” which is 708 (km of transect) multiplied by the strip width ($B=150 = 0.15$ km) multiplied by 2.

```
library("R2WinBUGS")
data<- read.csv("Muntjac.csv")
nind<-nrow(data)
y<-rep(1,nind)
nz<-400
y<-c(y,rep(0,nz))
x<-data[,3]
x<-c(x,rep(NA,nz))
z<-y
data<-list(y=y,x=x,nz=nz,nind=nind,B=150,striparea=708*.15*2)

cat("
model{
b~dunif(0,10)
psi~dunif(0,1)
```

```

2622 for(i in 1:(nind+nz)){
2623   z[i]~dbern(psi)    # DA Variables
2624   x[i]~dunif(0,B)    # B=strip width
2625   p[i]<-exp(logp[i])  # DETECTION MODEL
2626   logp[i]<- -((x[i]*x[i])*b)
2627   #logp[i]<- -b*log(x[i]+1)
2628   mu[i]<-z[i]*p[i]
2629   y[i]~dbern(mu[i])  # OBSERVATION MODEL
2630 }
2631 N<-sum(z[1:(nind+nz)])
2632 D<- N/striparea # area of transects
2633 }
2634 ",file="dsamp.txt")

```

2635 Next, we provide inits, indicate which parameters to monitor, and then pass
 2636 those things to WinBUGS:

```

2637 params<-list('b','N','D','psi')
2638 inits = function() {list(z=z, psi=runif(1), b=runif(1,0,.02) )}
2639 fit = bugs(data, inits, params, model.file="dsamp.txt",
2640 working.directory=getwd(),debug=T, n.chains=3, n.iter=4000, n.burnin=1000, n.thin=2)

```

2641 Posterior summaries are provided in the following table. Estimated density is pretty
 2642 low, 1.1 individuals per sq. km.¹²

```

2643 node mean sd MC error 2.5% median 97.5% start sample
2644 D 1.096 0.1694 0.009122 0.8098 1.078 1.474 501 4500
2645 N 232.8 35.99 1.938 172.0 229.0 313.0 501 4500
2646 b 5.678E-4 1.05E-4 4.129E-6 3.867E-4 5.616E-4 7.949E-4 501 4500
2647 deviance 681.2 16.72 0.7536 650.8 680.6 716.6 501 4500
2648 psi 0.5099 0.08238 0.004442 0.3681 0.5033 0.6918 501 4500

```

4.7 SUMMARY AND OUTLOOK

2649 Traditional closed population capture-recapture models are closely related to bino-
 2650 mial generalized linear models. Indeed, the only real distinction is that in capture-
 2651 recapture models, the population size parameter N (corresponding also to the size
 2652 of a hypothetical “complete” data set) is unknown. This requires special con-
 2653 sideration in the analysis of capture-recapture models. The classical approach to
 2654 inference recognizes that the observations don’t have a standard binomial distribu-
 2655 tion but, rather, a truncated binomial (from which which the so-called “conditional

¹² much lower than Samba’s : Observers walked about 708 km from 39 transects in Nagarahole and the muntjac density is about 3 per sq km.. I need to get to the bottom of this.

likelihood” derives) since we only have encounter frequency data on observed individuals. If instead we analyze the models using data augmentation, the observations can be modeled using a zero-inflated binomial distribution. In short, when we deal with the unknown-N problem using data augmentation then we are left with zero-inflated GLM and GLMMs instead of ordinary GLM or GLMMs. The analysis of such zero-inflated models is practically convenient, especially using the various Bayesian analysis packages that use the BUGS language.

Spatial capture-recapture models that we will consider in the rest of the chapters of this book are closely related to what have been called individual covariate models. Heuristically, spatial capture-recapture models arise by defining individual covariates based on observed locations of individuals – we can think of using some function of mean encounter location as an individual covariate. We did this in a novel way, by using distance to the centroid of the trapping array as a covariate. We analyzed the “full likelihood” using data augmentation, and placed a prior distribution on the individual covariate which was derived from an assumption that individual locations are, a priori, uniformly distributed in space. This assumption provides for invariance of the density estimator to the choice of population size area (induced by maximum distance from the centroid of the). The model addressed some important problems in the use of closed population models: it allows for heterogeneity in encounter probability due to the spatial context of the problem and it also provides a direct estimate of density because area is a feature of the model (via the prior on the individual covariate). The model is still not completely general because the model does not make use of the fully spatial encounter histories, which provide direct information about the locations and density of individuals. A specific individual covariate model that is in widespread use is classical “distance sampling.” The model underlying distance sampling is precisely a special kind of SCR model - but one without replicate samples. Understanding distance sampling and individual covariate models more broadly provides a solid basis for understanding and analyzing spatial capture-recapture models.

5

FULLY SPATIAL CAPTURE-RECAPTURE MODELS

6

FULLY SPATIAL CAPTURE-RECAPTURE MODELS

In previous sections we discussed some classes of models that could be viewed as primitive spatial capture-recapture models. We looked at a basic distance sampling model and we also considered a classical individual covariate modeling approach in which we defined a covariate to be the distance from (estimated) home range center to the center of the trap array. These were spatial in the sense that they included some characterization of where individuals live but, on the other hand, only a primitive or no characterization of trap location. That said, very little distinguishes these two models from spatial capture-recapture models that we consider in this chapter which fully recognize the spatial attribution of both individual animals *and* the locations of encounter devices.

Fully spatial capture-recapture models must accommodate the spatial organization of individuals and the encounter devices because the encounter process occurs at the level of individual traps. Failure to consider the trap-specific collection of data is the key deficiency with classical ad-hoc approaches which aggregate encounter information to the resolution of the entire trap array. We have seen previously some problems that this induces - imbalance in trap-level effort over time is problematic, and not being able to deal with trap-specific behavioral responses. Here, we resolve that by developing what is basically an individual covariate model but operating at the level of traps. That is, we develop our first fully spatial capture-recapture model which turns out to be precisely the model considered in section 3.XXX but instead of defining the individual covariate to be distance to centroid of the array we define J individual covariates - the distance to *each* trap. And, instead of using estimates of individual locations \mathbf{s} , we consider a fully hierarchical model in which we regard \mathbf{s} as a latent variable and impose a prior distribution on

it. We can think of having J independent capture-recapture studies generating one data set for each trap, and applying the individual covariate model with random activity centers, and that is all the basic SCR model is.

In the following sections of this chapter we investigate the basic spatial capture-recapture model and address some important considerations related to its analysis in **WinBUGS**. We also demonstrate how to summarize posterior output for the purposes of producing density maps or spatial predictions of density.

6.1 SAMPLING DESIGN AND DATA STRUCTURE

In our development here, we will assume a standard sampling design in which an array of J traps is operated for K time periods (say, nights) producing encounters of n individuals. Because sampling occurs by traps and also over time, the most general data structure yields encounter histories for *each individual* that are temporally *and* spatially indexed. Thus a typical data set will include an encounter history *matrix* for each individual. For the most basic model, there are no time-varying covariates that influence encounter, there are no explicit individual-specific covariates, and there are no covariates that influence density we will develop models in this chapter for encounter data that are aggregated over the temporal replicates. For example, suppose we observe 6 individuals in sampling at 4 traps over 3 nights of sampling then a plausible data set is the 6×4 matrix of encounters, out of 3, of the form:

	trap1	trap2	trap3	trap4
[1,]	1	0	0	0
[2,]	0	2	0	0
[3,]	0	0	0	1
[4,]	0	1	0	0
[5,]	0	0	1	1
[6,]	1	0	1	0

We develop models in this chapter for devices such as “hair snares” or other DNA sampling methods (Kéry et al., 2010; Gardner et al., 2010) and related types of sampling problems so that we can suppose that “traps” may capture any number of individuals and an individual may be captured in any number of traps during each occasion but individuals can be encountered at most 1 time in a trap during any occasion. Thus, this is a “multi-catch” type of sampling (?, p. xyz). The statistical assumptions are that individual encounters within and among traps are independent. These basic (but admittedly at this point somewhat imprecise) assumptions define the basic spatial capture-recapture model, which we will refer to as “SCR0” henceforth¹ so that we may use that model as a point of reference

¹RC: It would be nice to have a running series of figures to display the various types of models. Each figure could have the same set of traps, use the same symbols, etc... It’s probably worth showing example data (and latent variables) in a table too

without having to provide a long-winded enumeration of assumptions and sampling design each time we do. We will make things more precise as we develop a formal statistical definition of the model shortly.

While the model is mostly directly relevant for hair snares and other DNA sampling methods for which multiple detections of an individual are not distinguishable, we will also make use of the model for data that arise from camera-trapping studies. In practice, with camera trapping, individuals might be photographed several times in a night but we will typically distill such data into a single binary encounter event for reasons discussed later in chapter 6.

6.2 THE BINOMIAL OBSERVATION MODEL

We assume that the individual and trap-specific encounters, y_{ij} , are mutually independent outcomes of a binomial random variable:

$$y_{ij} \sim \text{Bin}(K, p_{ij}) \quad (6.2.1)$$

This is the basic model underlying “logistic regression” (chapter 2) as well as standard closed population models (chapter 3). The key element of the model is that the encounter probability p_{ij} is indexed by (i.e., depends on) both individual and trap. In a sense, then, we can think of each *trap* as producing individual level encounter history data of the classical variety - an $n_{\text{ind}} \times n_{\text{rep}}$ matrix of 0's and 1's (this is the “encountered at most 1 time” assumption).

As we did in section XXX.YYY, we will make explicit the notion that p_{ij} is defined conditional on “where” individual i lives. Naturally, we think about defining an individual home range and then relating p_{ij} explicitly to the centroid of the individual's home range, or its center of activity (Efford, 2004; Borchers and Efford, 2008; Royle and Young, 2008). Therefore, define \mathbf{s}_i , a two-dimensional spatial coordinate, to be the activity center for individual i . Then, the basic SCR model postulates that encounter probability, p_{ij} , is related by a decreasing function to distance between trap j , having location \mathbf{x}_j , and \mathbf{s}_i . Naturally, if we think of modeling binomial counts using logistic regression, we might specify the model according to:

$$\text{logit}(p_{ij}) = \alpha_0 + \theta * ||\mathbf{s}_i - \mathbf{x}_j|| \quad (6.2.2)$$

where, here, $||\mathbf{s}_i - \mathbf{x}_j||$ is the distance between \mathbf{s}_i and \mathbf{x}_j . We sometimes write $||\mathbf{s}_i - \mathbf{x}_j|| = \text{dist}(\mathbf{s}_i, \mathbf{x}_j) = d_{ij}$. Alternatively, if we think about distance sampling then we might use the “half-normal” model of the form:

$$p_{ij} = p_0 * \exp(-\theta * ||\mathbf{s}_i - \mathbf{x}_j||^2)$$

Or any of a large number of standard detection models that are commonly used (we consider more in chapter XYZ). The half-normal model implies

$$\log(p_{ij}) = \log(p_0) - \theta * ||\mathbf{s}_i - \mathbf{x}_j||^2 \quad (6.2.3)$$

2785 Whatever model encounter probability we choose, we should always keep in mind
 2786 that the model is described conditional on \mathbf{s}_i , which is an unobserved random
 2787 variable. Thus, to be precise about this, we should write the observation model as

$$y_{ij}|\mathbf{s}_i \sim \text{Bin}(K, p(\mathbf{s}_{ij}; \theta))$$

2788 Note that we probably expect that the parameter θ in Eq. 6.2.2 or 6.2.3 should
 2789 be negative, so that the probability of encounter decreases with distance between the
 2790 trap and individual home range center. The joint likelihood for the data, conditional
 2791 on the collection of individual activity centers, can therefore be expressed as

$$\mathcal{L}(\theta|\{\mathbf{y}_i, \mathbf{s}_i\}_{i=1}^N) = \prod_i \prod_j \text{Bin}(y_{ij}|p_{ij}(\theta))$$

2792 Which, if we switch the indices on the product operators, this shows the SCR
 2793 likelihood (conditional on \mathbf{s}) to be the product of J *independent* capture-recapture
 2794 likelihoods - one for each trap. However, the data have a “repeated measures” type
 2795 of structure, with each of the j likelihood contributions for each individual being
 2796 grouped by individual. Thus, we cannot analyze the model meaningfully by J trap-
 2797 specific models. In classical repeated measures types of models, we accommodate
 2798 the group structure of the data using random effects (random individual or group
 2799 level variables). For SCR models we take the same basic approach, which we develop
 2800 subsequently.

2801 6.2.1 Distance as a latent variable

2802 If we knew precisely every \mathbf{s}_i in the population (and how many, N), then the model
 2803 specified by eqs. 6.2.1 and 6.2.2 or 6.2.3 is just an ordinary logistic regression type
 2804 of a model which we learned how to fit using **WinBUGS** previously (chapt. 2),
 2805 with a covariate d_{ij} . However, the activity centers are unobservable even in the best
 2806 possible circumstances. In that case, d_{ij} is an unobserved variable, analogous to
 2807 classical “random effects” models. We need to therefore extend the model to accom-
 2808 modate these random variables with an additional model component. A standard,
 2809 and perhaps not unreasonable, assumption is the so-called “uniformity assumption”
 2810 which is to say that the \mathbf{s}_i are uniformly distributed over space (the obvious next
 2811 question “which space?” is addressed below). This uniformity assumption amounts
 2812 to a uniform prior distribution on \mathbf{s}_i , i.e., the pdf of \mathbf{s}_i is constant, which we may
 2813 express

$$\Pr(\mathbf{s}_i) \text{propto} \text{const} \tag{6.2.4}$$

2814 To summarize the preceeding model developing, a basic SCR model is defined
 2815 by 3 essential components:

- 2816 (1) Observation model: $y_{ij}|\mathbf{s}_i \sim \text{Bin}(K, p_{ij})$
- 2817 (2) Encounter probability: $\text{logit}(p_{ij}) = \alpha_0 + \theta * \|\mathbf{s}_i - \mathbf{x}_j\|$

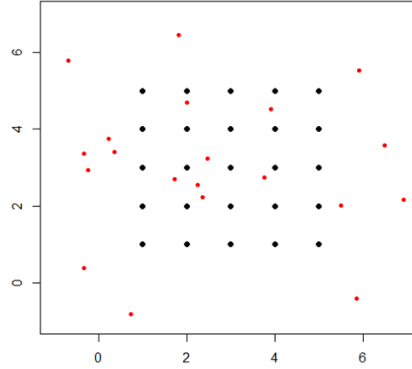


Figure 6.1. Realization of a binomial point process

2818 (3) Point process model: $\Pr[\mathbf{s}_i] \propto \text{const}$

2819 Therefore, the SCR model is little more than an ordinary capture-recapture model
 2820 for closed populations. It is such a model, but augmented with a set of “individual
 2821 effects”, \mathbf{s}_i , which relate some sense of individual location to encounter probability.
 2822 As it turns out, assumption (3) is usually not precise enough to fit a model in
 2823 practice for reasons we discuss in the following section. We will give another way to
 2824 represent this prior distribution that is more concrete, but it depends on specifying
 2825 the “state-space” of the random variable \mathbf{s}_i . The term “state-space” is a technical
 2826 way of saying “possible outcomes”.

6.3 THE BINOMIAL POINT-PROCESS MODEL

2827 The collection of individual activity centers $\mathbf{s}_1, \dots, \mathbf{s}_N$ represent a realization of a
 2828 *binomial point process* (Illian, 2008a, p. xyz). The binomial point process (BPP)
 2829 is analogous to a Poisson point process in the sense that it represents a “random
 2830 scatter” of points in space - except that the total number of points is *fixed*, whereas,
 2831 in a Poisson point process it is random (having a Poisson distribution). As an
 2832 example, we show in Fig. 6.1 locations of 20 individual activity centers (black
 2833 dots) in relation to a grid of 25 traps. For a Poisson point process the number of
 2834 such points in the prescribed state-space would be random whereas often we will
 2835 simulate fixed numbers of points, e.g., for evaluating the performance of procedures
 2836 such as how well does our estimator perform of $N = 50$?

It is natural to consider a binomial point process in the context of capture-recapture models because it preserves N in the model and thus preserves the linkage directly with closed population models. In fact, under the binomial point process model then Model M0 and other closed models are simple limiting cases of SCR models. In addition, use of the BPP model allows us to use data augmentation for Bayesian analysis of the models as in chapter 3, thus yielding a methodologically coherent approach to analyzing the different classes of models. Despite this, making explicit assumptions about N , such as Poisson, is convenient in some cases (see chapt. XYZ).

One consequence of having fixed N , in the BPP model, is that the model is not strictly a model of “complete spatial randomness”. This is because if one forms counts $n(A_1), \dots, n(A_k)$ in any set of disjoint regions say A_1, \dots, A_k , then these counts are *not* independent. In fact, they have a multinomial distribution (see Illian, 2008a, p. XYZ). Thus, the BPP model introduces a slight bit of dependence in the distribution of points. However, in most situations this will have no practical effect on any inference or analysis and, as a practical matter, we will usually regard the BPP model as one of spatial independence among individual activity centers because each activity center is distributed independently of each other activity center. Despite this implicit independence we see in Fig. 6.1 that realizations of randomly distributed points will typically exhibit distinct non-uniformity. Thus, independent, uniformly distributed points will almost never appear regularly, uniformly or systematically distributed. For this reason, the basic binomial (or Poisson) point process models are enormously useful in practical settings. More relevant for SCR models is that we actually have a little bit of data for some individuals and thus the resulting posterior point pattern can deviate strongly from uniformity (we should note this elsewhere too). The uniformity hypothesis is only a *prior* distribution which is directly affected by the quantity and quality of observations.

6.3.1 Definition of home range center

Some will be offended by our use of the concept of “home range center” and thus will have difficulty in believing that the resulting model is really useful for anything. Indeed, the idea of a home range or activity center is a vague concept anyway, a purely phenomenological construct. Despite this, it doesn’t really matter whether or not a home range makes sense for a particular species - individuals of any species inhabit *some* region of space and we can define the “home range center” to be the center of the space that individual was occupying (or using) during the period in which traps were active. Thinking about it in that way, it could even be observable (almost) as the centroid of a very large number of radio fixes over the course of a survey period or a season. Thus, this practical version of a home range center is a well-defined construct regardless of whether one thinks the home range concept is meaningful, even if individuals are not particularly territorial. This is why we usually use the term “activity center” or maybe even “centroid of space usage”

and we recognize that this construct is a transient thing which applies only to a well-defined period of study.

6.3.2 The state-space of the point process

Shortly we will focus on Bayesian analysis of this model with N known so that we can directly apply what we learned in chapter 2 to this situation. To do this, we note that the individual effects $\mathbf{s}_1, \dots, \mathbf{s}_N$ are unknown quantities and we will need to be able to simulate each \mathbf{s}_i in the population from the posterior distribution. It should be self-evident that we cannot simulate the \mathbf{s}_i unless we describe precisely the region over which those \mathbf{s}_i 's are uniformly distributed. This is the quantity referred to above as the state-space, denoted henceforth by \mathcal{S} , which is a region or a set of points comprising the potential values of \mathbf{s}_i . Thus, an equivalent explicit statement of the “uniformity assumption” is

$$\mathbf{s}_i \sim \text{Unif}(\mathcal{S})$$

Prescribing the state-space

Evidently, we need to define the state-space, \mathcal{S} . How can we possibly do this objectively? Prescribing any particular \mathcal{S} seems like the equivalent of specifying a “buffer” which we criticized previously as being ad hoc. How is it that choosing a state-space is *not* ad hoc? As a practical matter, it turns out that estimates of density are insensitive to choice of the state-space. As we observed in chapter 11, it is true that N increases with \mathcal{S} , but only at the same rate as \mathcal{S} under the prior assumption of constant density. As a result, we say that density is invariant to \mathcal{S} as long as \mathcal{S} is sufficiently large. Thus, while choice of \mathcal{S} is (or can be) essentially arbitrary, once \mathcal{S} is chosen, it defines the population being exposed to sampling, which scales appropriately with the size of the state-space.

For our simulated system developed previously in this chapter, we defined the state space to be a square within which our traps were centered perfectly. For many practical situations this might be an acceptable approach to defining the state-space. We provide an example of this in section 6.7 below in which the trap array is irregular and also situated within a realistic landscape that is distinctly irregular. In general, it is most practical to define the state-space as a regular polygon (e.g., rectangle) containing the trap array without differentiating unsuitable habitat. Although defining the state-space to be a regular polygon has computational advantages (e.g., we can implement this more efficiently in **WinBUGS** and cannot for irregular polygons), a regular polygon induces an apparent problem of admitting into the state-space regions that are distinctly non-habitat (e.g., oceans, large lakes, ice fields, etc.). It is difficult to describe complex sets in mathematical terms that can be admitted to this spatial model. As an alternative, we can provide a representation of the state-space as a discrete set of points (section 6.9) that will allow specific points to be deleted or not depending on whether they represent habitat, or we can define the state-space as an intersection of polygons, and analysis of

models with state-space defined in that way can be analyzed easily using MCMC (see section XYZ in chapt. 6). In what follows below we provide an analysis of the camera data defining the state-space to be a regular continuous polygon (a rectangle).

6.3.3 Invariance and the State-space as a model assumption

We will assert for all models we consider in this book that density is invariant to the size and extent of \mathcal{S} , if \mathcal{S} is sufficiently large. In fact, this only holds as long as our model relating p_{ij} to \mathbf{s}_i is a decreasing function of distance. We can prove this thinking about a 1-d case where $E[y]$ for the “last cell” (i.e., for $d > B$ for B large enough) is 0. So it always contributes nothing to the likelihood, i.e., $E[n(\text{lastcell})] = 0$. [sketch out a proof of this], in regular situations in which the detection function decays monotonically with distance and prior density is constant. Sometimes our estimate of density can be influenced if we make \mathcal{S} too small but this might be sensible if \mathcal{S} is naturally well-defined. As we discussed in chapter 1, **choice of \mathcal{S} is part of the model and thus it makes sense that estimates of density might be sensitive to its definition in problems where it is natural to restrict \mathcal{S} .** One could imagine however that in specific cases where you’re studying a small population with well-defined habitat preferences that a problem could arise because changing the state-space around based on differing opinions and GIS layers really changes the estimate of total population size. But this is a real biological problem and a natural consequence of the spatial formalization of capture-recapture models - a feature, not a bug or some statistical artifact - and it should be resolved with better information and research, and not some arbitrary statistical artifact. For situations where there is not a natural choice of \mathcal{S} , we should default to choosing \mathcal{S} to be very large in order to achieve invariance or otherwise evaluate sensitivity of density estimates by trying a couple of different values of \mathcal{S} . This is a standard “sensitivity to prior” argument that Bayesians always have to be conscious of. We demonstrate this in our analysis of section 6.7 below. Note that $area(\mathcal{S})$ affects data augmentation. If you increase $area(\mathcal{S})$ then there are more individuals to account for and therefore the size of the augmented data set M must increase.

We have been told that one can carry-out non-Bayesian analyses of SCR models without having to specify the state-space of the point process or perhaps while only specifying it imprecisely. This assertion is incorrect. We assume people are thinking this because *they* don’t have to specify it explicitly because someone else has done it for them in a package that does integrated likelihood. Even to do integrated likelihood (see chapter 9) we have to integrate the conditional-on- \mathbf{s} likelihood over some 2-dimensional space. It might work that the integration can be done from $-\infty$ to $+\infty$ but that is a mathematical artifact of specific detection functions, and an implicit definition of a state-space that doesn’t make biological sense, even though it may in fact be innocuous;

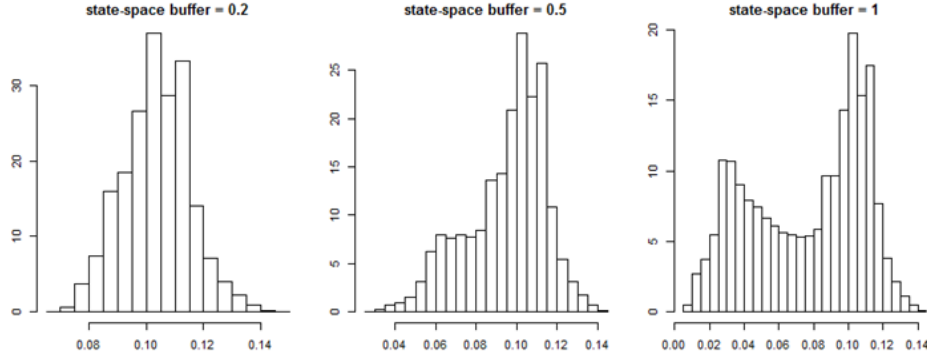


Figure 6.2. Needs a caption

6.3.4 Connection to Model Mh

SCR models are closely related to heterogeneity models. In SCR models, heterogeneity in encounter probability is induced by both the effect of distance in the model for detection probability and also from specification of the state-space. Clearly then the state-space is explicitly part of the model. To understand this, we have a random effect with some prior distribution:

$$\mathbf{s} \sim \text{uniform}(\mathcal{S})$$

And $p(\mathbf{s}) = p(y = 1|\mathbf{s})$ is some function of \mathbf{s} . Therefore, for any specific $g(p)$ and \mathcal{S} we can work out what the implied heterogeneity model is for example, the mean, variance or other moments of the population distribution of p can be evaluated by integrating $p(\mathbf{s})$ over the state-space of \mathbf{s} . Obviously the choice of $p(\mathbf{s})$ and the choice of \mathcal{S} interact to determine the effective heterogeneity in p . We show an illustration in Fig. 6.2 below which shows a histogram of p for a hypothetical population of 100000 individuals on a state-space enclosing our 5×5 trap array above, under the logistic model for distance. **R** code is provided in the **R** package **scrbook** to produce this analysis for the logistic and half-normal models. The histogram shows the encounter probability under buffers of 0.2, 0.5 and 1.0. We see the mass shifts to the left as the buffer increases, implying more individuals in the population but with lower encounter probability as their home range centers increase in distance from the trap array.

Another way to understand this is by representing \mathcal{S} as a set of discrete points on a grid. In the coarsest possible case where \mathcal{S} is a single arbitrary point, then every individual has exactly the same p . As we increase the number of points in \mathcal{S} then more distinct values of p are possible. As such, when \mathcal{S} is characterized by discrete points then SCR models are precisely a type of finite-mixture model (Norris III and

Pollock, 1996; Pledger, 2000), except where we have some information about which group an individual belong (i.e., where their activity center is), as a result of their captures in traps.

This context suggests the problem raised by Link (2003). He showed that in most practical situations N may not be identifiable across classes of mixture distributions which in the context of SCR models is the pair (g, \mathcal{S}) . The difference, however, is that we do obtain some direct information about \mathbf{s} in SCR models and therefore N is identifiable across models characterized by (g, \mathcal{S}) .

6.3.5 Connection to Distance Sampling

It is worth emphasizing that the basic SCR model is a binomial encounter model in which distance is a covariate. As such, it is striking similarity to a classical distance sampling model. Both have distance as a covariate but in classical distance sampling problems the focus is on the distance between the observer and the animal at an instant in time, not the distance between a trap and an animal's home range center. Thus in distance sampling, "distance" is *observed* for those individuals that appear in the sample. Conversely, in SCR problems, it is only imperfectly observed (we have partial information in the form of trap observations). Clearly, it is preferable to observe distance if possible, but as we will discuss in chapter XYZ, distance sampling requires field methods that are often not practical in many situations, e.g. when surveying tigers. Furthermore, SCR models allow us to relax many of the assumption made in classical distance sampling, and SCR models allow for estimates of quantities other than density, such as home range size.

6.4 SIMULATING SCR DATA

It is always useful to simulate data because it allows you to understand the system that you're modeling and also calibrate your understanding with the parameter values of the model. That is, you can simulate data using different parameter values until you obtain data that "looks right" based on your knowledge of the specific situation that you're interested in. Here we provide a simple script to illustrate how to simulate spatial encounter history data. In this exercise we simulate data for 100 individuals and a 25 trap array laid out in a 5×5 grid of unit spacing. The specific encounter model is the half-normal model given above and we used this code to simulate data used in subsequent analyses. The 100 activity centers were simulated on a state-space defined by a 8×8 square within which the trap array was centered (thus the trap array is buffered by 2 units). Therefore, the density of individuals in this system is fixed at $100/64$.

```
set.seed(2013)
# create 5 x 5 grid of trap locations with unit spacing
traplocs<- cbind(sort(rep(1:5,5)),rep(1:5,5))
```



```

3019 Dmat<-e2dist(traplocs,traplocs) # in cases where speed doesn't matter, it might be
3020                                     # clearer to just show the slow for-loop.
3021                                     # Plus, people will want to copy/paste this stuff
3022 ntraps<-nrow(traplocs)
3023
3024 # define state-space of point process. (i.e., where animals live).
3025 # "delta" just adds a fixed buffer to the outer extent of the traps.
3026 delta<-2
3027 Xl<-min(traplocs[,1] - delta)
3028 Xu<-max(traplocs[,1] + delta)
3029 Yl<-min(traplocs[,2] - delta)
3030 Yu<-max(traplocs[,2] + delta)
3031
3032 N<-100    # population size
3033 K<- 20    # number nights of effort
3034
3035 sx<-runif(N,Xl,Xu)    # simulate activity centers
3036 sy<-runif(N,Yl,Yu)
3037 S<-cbind(sx,sy)
3038 D<- e2dist(S,traplocs) # distance of each individual from each trap
3039
3040 alpha0<- -2.5        # define parameters of encounter probability
3041 sigma<- 0.5          #
3042 theta<- 1/(2*sigma*sigma)
3043 probcap<- expit(-2.5)*exp( - theta*D*D)    # probability of encounter
3044 # now generate the encounters of every individual in every trap
3045 Y<-matrix(NA,nrow=N,ncol=ntraps)
3046 for(i in 1:nrow(Y)){
3047     Y[i,<-rbinom(ntraps,K,probcap[i,])
3048 }

```

Subsequently we will generate data using this code packaged in an R function called `simSCR0.fn` which takes a number of arguments including `discard0` which, if `TRUE`, will return only the encounter histories for captured individuals. A second argument is `array3d` which, if `TRUE`, returns the 3-d encounter history array instead of the aggregated `nind × ntraps` encounter frequencies (see below). Finally we provide a random number seed, `sd` which we always set to 2013 in our analyses. Thus we obtain a data set as above using the following command

```

3056 data<-simSCR0.fn(discard0=TRUE,array3d=FALSE,sd=2013)

```

The **R** object `data` is a list, so let's take a look at what's in the list and then harvest some of its elements for further analysis below.

```

3059 > names(data)
3060 [1] "Y"          "traplocs" "xlim"      "ylim"      "N"          "alpha0"    "beta"
3061 [8] "sigma"      "K"

```

```

3062 > Y<-data$Y
3063 > traplocs<-data$traplocs

```

3064 6.4.1 Formatting and manipulating real data sets

3065 Conventional capture-recapture data are easily stored and manipulated as a 2-
 3066 dimensional array, an $nind \times nperiod$ matrix, which is maximally informative for
 3067 any conventional capture-recapture model, but not for spatial capture-recapture
 3068 models. For SCR models we must preserve the spatial information in the encounter
 3069 history information. We will routinely analyze data from 3 standard formats:

- 3070 (1) The basic 2-dimensional data format, which is an $nind \times ntraps$ encounter
 3071 frequency matrix such as that simulated previously;
- 3072 (2) The maximally informative 3-dimensional array which we establish here the
 3073 convention that it has dimensions $nind \times nperiods \times ntraps$ and
- 3074 (3) We use a compact format - the “SCR flat format” - which we describe below
 3075 in section 6.7.

3076 To simulate data in the most informative format - the “3-d array” - we can use the
 3077 **R** commands given previously but replace the last 4 lines with the following:

```

3078 Y<-array(NA,dim=c(N,K,ntraps))
3079 for(i in 1:nrow(Y)){
3080   for(j in 1:ntraps){
3081     Y[i,1:K,j]<-rbinom(K,1,probcap[i,j])
3082   }
3083 }

```

3084 We see that a collection of K binary encounter events are generated for *each*
 3085 individual and for *each* trap. The probabilities have those Bernoulli trials are
 3086 computed based on the distance from each individuals home range center and the
 3087 trap (see calculation above), and those are housed in the matrix probcap. Our
 3088 data simulator function `simSRC0.fn` will return the full 3-d array if `array3d=TRUE`
 3089 is specified in the function call. To recover the 2-d matrix from the 3-d array, and
 3090 subset the 3-d array to individuals that were captured, we do this:

```

3091 Y2d<- apply(Y,c(1,3),sum) # sum over the ‘‘replicates’’ dimension (2nd margin of the array)
3092 ncaps<-apply(Y2d,1,sum)   # compute how many times each individual was captured
3093 Y<-Y[ncaps>0,,]          # keep those individuals that were captured

```

6.5 FITTING AN SCR MODEL IN BUGS

3094 Clearly if we somehow knew the value of N then we could fit this model directly
 3095 because, in that case, it is a special kind of logistic regression model - one with a
 3096 random effect, but that enters into the model in a peculiar fashion - and also with
 3097 a distribution (uniform) which we don’t usually think of as standard for random

effects models. So our aim here is to analyze the known- N problem, using our simulated data, as an incremental step in our progress toward fitting more generally useful models.

To begin, we use our simulator to grab a data set and then harvest the elements of the resulting object for further analysis.

```

3103 data<-simSCR0.fn(discard0=FALSE,sd=2013)
3104 y<-data$Y
3105 traplocs<-data$traplocs
3106 nind<-nrow(y)
3107 X<-data$traplocs
3108 J<-nrow(X)
3109 y<-rbind(y,matrix(0,nrow=(100-nrow(y)),ncol=J ) )
3110 Xl<-data$xlim[1]
3111 Yl<-data$ylim[1]
3112 Xu<-data$xlim[2]
3113 Yu<-data$ylim[2]

```

Note that we specify `discard0 = FALSE` so that we have a "complete" data set, i.e., one with the all-zero encounter histories corresponding to uncaptured individuals. Now, within an **R** session, we can create the **BUGS** model file and fit the model using the following commands. This model describes the half-normal detection model but it would be trivial to modify that to various others including the logistic described above. One consequence of using the half-normal is that we have to constrain the encounter probability to be in $[0, 1]$ which we do here by defining `alpha0` to be the logit of the intercept parameter `p0`. Note that the distance covariate is computed within the **BUGS** model specification given the matrix of trap locations, `X`, which is provided to **WinBUGS** as data.

```

3124 cat("
3125 model {
3126   alpha0~dnorm(0,.1)
3127   logit(p0)<- alpha0
3128   theta~dnorm(0,.1)
3129   for(i in 1:N){
3130     s[i,1]~dunif(Xl,Xu)
3131     s[i,2]~dunif(Yl,Yu)
3132     for(j in 1:J){
3133       d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
3134       y[i,j] ~ dbin(p[i,j],K)
3135       p[i,j]<- p0*exp(- theta*d[i,j]*d[i,j])
3136     }
3137   }
3138 }
3139 }
3140 ",file = "SCR0a.txt")

```

Next we do a number of organizational activities including bundling the data for **WinBUGS**, defining some initial values, the parameters to monitor and some basic MCMC settings. We choose initial values for the activity centers **s** by generating uniform random numbers in the state-space but, for the observed individuals, we replace those values by each individual's mean trap coordinate for all encounters

```

3146 sst<-cbind(runif(nind,Xl,Xu),runif(nind,Yl,Yu)) # starting values for s
3147 for(i in 1:nind){
3148   if(sum(y[i,])==0) next
3149   sst[i,1]<- mean( X[y[i,]>0,1] )
3150   sst[i,2]<- mean( X[y[i,]>0,2] )
3151 }
3152
3153 data <- list (y=y,X=X,K=K,N=nind,J=J,Xl=Xl,Yl=Yl,Xu=Xu,Yu=Yu)
3154 inits <- function(){
3155   list (alpha0=rnorm(1,-4,.4),theta=runif(1,1,2),s=sst)
3156 }
3157
3158 library("R2WinBUGS")
3159 parameters <- c("alpha0","theta")
3160 nthin<-1
3161 nc<-3
3162 nb<-1000
3163 ni<-2000
3164 out <- bugs (data, inits, parameters, "SCROa.txt", n.thin=nthin,
3165   n.chains=nc, n.burnin=nb,n.iter=ni,debug=TRUE,working.dir=getwd())

```

There is little to say about the preceding basic operations other than to suggest that the interested reader explore the output and additional analyses by running the script provided in the **R** package **scrbook**. We ran 1000 burn-in and 1000 after burn-in, 3 chains, to obtain 3000 posterior samples. Because we know N for this particular data set we only have 2 parameters of the detection model to summarize (**alpha0** and **theta**). When the object **out** is produced we print a summary of the results as follows:

```

3173 > print(out,digits=3)
3174 Inference for Bugs model at "SCROa.txt", fit using WinBUGS,
3175 3 chains, each with 2000 iterations (first 1000 discarded)
3176 n.sims = 3000 iterations saved
3177
3178      mean      sd    2.5%    25%    50%    75%   97.5%  Rhat n.eff
3179 alpha0  -2.496  0.224  -2.954  -2.648  -2.48  -2.340  -2.091  1.013   190
3179 theta    2.442  0.419   1.638   2.145   2.44   2.721   3.303  1.005   530
3180 deviance 292.803 21.155 255.597 277.500 291.90 306.000 339.302 1.006   380
3181
3182 For each parameter, n.eff is a crude measure of effective sample size,
3183 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
3184
3185 DIC info (using the rule, pD = Dbar-Dhat)

```

3186 `pD = -138.8 and DIC = 154.0`
 3187 `DIC is an estimate of expected predictive error (lower deviance is better).`

3188 We know the data were generated with `alpha0 = -2.5` and `theta = -2`. The
 3189 estimates look reasonably close to those data-generating values and we probably feel
 3190 pretty good about the performance of the Bayesian analysis and MCMC algorithm
 3191 that WinBUGS cooked-up based on our sample size of 1 data set. It is worth noting
 3192 that the Rhat statistics indicate reasonable convergence but, as a practical matter,
 3193 we might choose to run the MCMC algorithm for additional time to bring these
 3194 closer to 1.0 and to increase the effective posterior sample size (`n.eff`). Other
 3195 summary output includes “deviance” and related things including the deviance
 3196 information criterion (DIC). We discuss these things in chapter XXXX.

6.6 UNKNOWN N

3197 In all real applications N is unknown and that fact is kind of an important feature
 3198 of the capture-recapture problem! We handled this important issue in chapter 3
 3199 using the method of data augmentation which we apply here to achieve a realistic
 3200 analysis of Model SCR0. As with the basic closed population models considered
 3201 previously, we formulate the problem here by augmenting our observed data set
 3202 with a number of “all zero” encounter histories - what we referred to in Chapter
 3203 3 as potential individuals. If n is the number of observed individuals, then let
 3204 $M - n$ be the number of potential individuals in the data set. For the basic y_{ij}
 3205 data structure (individuals x traps encounter frequencies) we simply add additional
 3206 rows of “all 0” observations to that data set. This is because such “individuals” are
 3207 unobserved, and therefore necessarily have $y_{ij} = 0$ for all j . A data set, say with 4
 3208 traps and 6 individuals, augmented with 4 pseudo-individuals therefore might look
 3209 like this:

	trap1	trap2	trap3	trap4
[1,]	1	0	0	0
[2,]	0	2	0	0
[3,]	0	0	0	1
[4,]	0	1	0	0
[5,]	0	0	1	1
[6,]	1	0	1	0
[7,]	0	0	0	0
[8,]	0	0	0	0
[9,]	0	0	0	0
[10,]	0	0	0	0

3221 We typically have more than 4 traps and, if we’re fortunate, many more indi-
 3222 viduals in our data set.

3223 For the augmented data, we introduce a set of binary latent variables (the data
 3224 augmentation variables), z_i , and the model is extended to describe $\Pr(z_i = 1)$
 3225 which is, in the context of this problem, the probability that an individual in the

augmented data set is a member of the population that was sampled. In other words, if $z_i = 1$ for one of the “all zero” encounter histories, this is implied to be a sampling zero whereas observations for which $z_i = 0$ are “structural zeros” under the model.

How big does the augmented data set have to be? We discussed this issue in chapt. 3 where we noted that the size of the data set is equivalent to the upper limit of a uniform prior distribution on N . Practically speaking, it should be sufficiently large so that the posterior distribution for N is not truncated. On the other hand, if it is too large then unnecessary calculations are being done. An approach to choosing M by trial-and-error is indicated. You can take a ballpark estimate of the probability that an individual is captured (at all during the study), obtain N as $n/pcap$, and then set $M = 2 * N$, as a first guess. Do a short MCMC run and then consider whether you need to do something different. See chapt. 10 for an example of this. Kery and Schaub (2011, ch. 6) provide an assessment of choosing M in closed population models.

Analysis by data augmentation removes N as an explicit parameter of the model. Instead, N is a derived parameter, computed by $N = \sum_{i=1}^M z_i$. Similarly, *density*, D , is also a derived parameter computed as $D = N/area(\mathcal{S})$. For our simulator, we’re using an 8×8 state-space and thus we will compute D as $D = N/64$.

6.6.1 Analysis using data augmentation in WinBUGS

As before we begin by obtaining a data set using our `simSCR0.fn` routine and then harvesting the required data objects from the resulting data list. Note that we use the `discard0=TRUE` option this time so that we get a “real” data set with no all-zero encounter histories. After harvesting the data we produce the **WinBUGS** model specification which now includes M encounter histories including the augmented potential individuals, the data augmentation parameters z_i , and the data augmentation parameter ψ .

```

data<-simSCR0.fn(discard0=TRUE,sd=2013)
y<-data$Y
traplocs<-data$traplocs
nind<-nrow(y)
X<-data$traplocs
J<-nrow(X)
Xl<-data$xlim[1]
Yl<-data$ylim[1]
Xu<-data$xlim[2]
Yu<-data$ylim[2]

cat("
model {
  alpha0~dnorm(0,.1)
  logit(p0)<- alpha0

```

```

3268 theta~dnorm(0,.1)
3269 psi~dunif(0,1)
3270
3271 for(i in 1:M){
3272   z[i] ~ dbern(psi)
3273   s[i,1]~dunif(Xl,Xu)
3274   s[i,2]~dunif(Yl,Yu)
3275   for(j in 1:J){
3276     d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
3277     y[i,j] ~ dbin(p[i,j],K)
3278     p[i,j]<- z[i]*p0*exp(- theta*d[i,j]*d[i,j])
3279   }
3280 }
3281 N<-sum(z[])
3282 D<-N/64
3283 }
3284 ",file = "SCR0a.txt")

```

3285 To prepare our data we have to augment the data matrix y with $M - n$ all-
 3286 zero encounter histories, we have to create starting values for the variables z_i and
 3287 also the activity centers s_i of which, for each, we require M values. Otherwise the
 3288 remainder of the code for bundling the data, creating initial values and executing
 3289 **WinBUGS** looks much the same as before except with more or differently named
 3290 arguments.

```

3291 ## Data augmentation stuff
3292 M<-200
3293 y<-rbind(y,matrix(0,nrow=M-nind,ncol=ncol(y)))
3294 z<-c(rep(1,nind),rep(0,M-nind))
3295
3296 sst<-cbind(runif(M,Xl,Xu),runif(M,Yl,Yu)) # starting values for s
3297 for(i in 1:nind){
3298   if(sum(y[i,])==0) next
3299   sst[i,1]<- mean( X[y[i,]>0,1] )
3300   sst[i,2]<- mean( X[y[i,]>0,2] )
3301 }
3302 data <- list (y=y,X=X,K=K,M=M,J=J,Xl=Xl,Yl=Yl,Xu=Xu,Yu=Yu)
3303 inits <- function(){
3304   list (alpha0=rnorm(1,-4,.4),theta=runif(1,1,2),s=sst,z=z)
3305 }
3306
3307 library("R2WinBUGS")
3308 parameters <- c("alpha0","theta","N")
3309 nthin<-1
3310 nc<-3
3311 nb<-1000
3312 ni<-2000

```

```

3313 out <- bugs (data, inits, parameters, "SCROa.txt", n.thin=nthin,n.chains=nc,
3314   n.burnin=nb,n.iter=ni,debug=TRUE,working.dir=getwd())

```

3315 **Remarks:** (1) Note the differences in this new **WinBUGS** model with that
 3316 appearing in the known- N version. (2) Also the input data has changed - the
 3317 augmented data set has more rows of all-zeros. Previously we knew that $N = 100$
 3318 but in this analysis we pretend not to know N , but think that $N = 200$ is a good
 3319 upper-bound; (3) Population size $N(S)$ is a derived parameter, being computed by
 3320 summing up all of the data augmentation variables z_i (as we've done previously);
 3321 (4) Density, $D \equiv D(S)$, is also a derived parameter. Summarizing the output from
 3322 **WinBUGS** produces:

```

3323 > print(out1,digits=2)
3324 Inference for Bugs model at "SCROa.txt", fit using WinBUGS,
3325 3 chains, each with 2000 iterations (first 1000 discarded)
3326 n.sims = 3000 iterations saved
3327      mean      sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
3328 alpha0   -2.57  0.23   -3.04  -2.72  -2.56  -2.41  -2.15  1.01   320
3329 theta     2.46  0.42    1.63   2.16   2.46   2.73   3.33  1.02   120
3330 N        113.62 15.73   86.00 102.00 113.00 124.00 147.00 1.01   260
3331 D          1.78  0.25    1.34   1.59   1.77   1.94   2.30  1.01   260
3332 deviance 302.60 23.67 261.19 285.47 301.50 317.90 354.91 1.00  1400
3333
3334 For each parameter, n.eff is a crude measure of effective sample size,
3335 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
3336
3337 DIC info (using the rule, pD = var(deviance)/2)
3338 pD = 279.9 and DIC = 582.5
3339 DIC is an estimate of expected predictive error (lower deviance is better).

```

3340 The column labeled “MC error” is the Monte Carlo error - the error inherent in
 3341 the attempt to compute these posterior summaries by MCMC. It is desirable to run
 3342 the Markov chain algorithm long enough so as to reduce the MC error to a tolerable
 3343 level. What constitutes tolerable is up to the investigator. Certainly less than 1% is
 3344 called for. As a general rule, Rhat gets closer to 1 and MC error decreases toward 0
 3345 as the number of iterations increases. We see that the estimated parameters (α_0 and
 3346 θ) are comparable to the previous results obtained for the known- N case, and also
 3347 not too different from the data-generating values. The posterior of N overlaps the
 3348 data-generating value substantially with a mean of 113.62. To obtain these results
 3349 we fitted the true data-generating model, that based on the half-normal detection
 3350 model, to a single simulated data set. For fun and excitement we fit the *wrong*
 3351 model - that with the logistic-linear detection model - to the same data set. This is
 3352 easily achieved by modifying the **WinBUGS** model specification above, although
 3353 we provide the **R** script in the **R** package **scrbook**. Those results are given below.
 3354 We see that the estimate of N , the main parameter of interest, is very similar to
 3355 that obtained under the correct model, convergence is worse (as measured by Rhat)
 3356 which probably doesn't have anything to do with the model being wrong, and the

posterior deviance and DIC favor the correct model. We consider the use of DIC for carrying-out model selection in chapter 12.

```
> print(out2,digits=2)
Inference for Bugs model at "SCR0a.txt", fit using WinBUGS,
3 chains, each with 2000 iterations (first 1000 discarded)
n.sims = 3000 iterations saved
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
alpha0	-1.59	0.27	-2.16	-1.77	-1.58	-1.42	-1.07	1.05	60
beta	3.77	0.43	2.92	3.48	3.79	4.05	4.66	1.04	70
N	122.57	18.67	90.00	109.00	122.00	135.00	163.00	1.00	3000
D	1.92	0.29	1.41	1.70	1.91	2.11	2.55	1.00	3000
deviance	312.67	22.43	271.00	297.20	311.50	327.00	359.60	1.02	130

```
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 = var(deviance)/2)
pD = 247.5 and DIC = 560.1
DIC is an estimate of expected predictive error (lower deviance is better).
```

6.6.2 Use of other BUGS engines: JAGS

There are two other popular **BUGS** engines in widespread use: **OpenBUGS** (Thomas et al., 2006) and **JAGS** (Plummer, 2003). Both of these are easily called from **R**. **OpenBUGS** can be used instead of **WinBUGS** by changing the package option in the bugs call to `package=OpenBUGS`. **JAGS** can be called using the function `jags()` in package **R2JAGS** which has nearly the same arguments as `bugs()`. We prefer to use the **R** library `rjags` (Plummer, 2009) which has a slightly different implementation that we demonstrate here as we reanalyze the simulated data set in the previous section (note: the same **R** commands are used to generate the data and package the data, inits and parameters to monitor). The function `jags.model` is used to initialize the model and run the MCMC algorithm for a period in which adaptive rejection (XXXX not sure XXXXX???) sampling is used. Then the Markov chains are updated using `coda.samples()` to obtain posterior samples for analysis, as follows:

```
jm<- jags.model("SCR0a.txt", data=data, inits=inits, n.chains=nc,
               n.adapt=nb))
jm<- coda.samples(jm, parameters, n.iter=ni-nb, thin=nthin)
```

We find that JAGS seems to be 20-30% faster for the basic SCR model which the reader can evaluate using the script `jags.winbugs.R` in the **R** package `scrbook`.

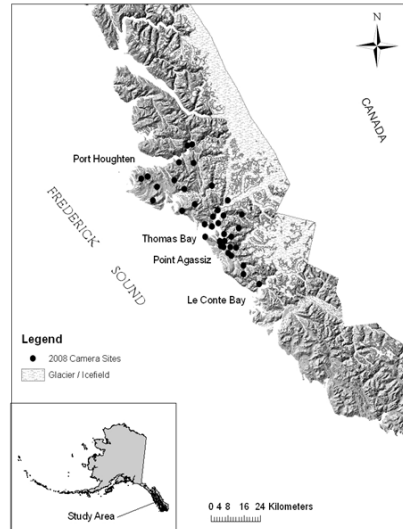


Figure 6.3. Wolverine camera trap locations from Magoun et al. (2011).

6.7 CASE STUDY: WOLVERINE CAMERA TRAPPING STUDY

We provide an analysis here of A. Magoun’s wolverine data (Magoun et al., 2011; Royle et al., 2011c). The study took place in SE Alaska (Fig. 6.3) where 37 cameras were operational for variable periods of time (min = 5 days, max = 108 days, median = 45 days). A consequence of this is that the binomial sample size K (see Eq. 6.2.1) is variable for each camera. Thus, we must provide a matrix of sample sizes as data to BUGS and modify the model specification in sec. 6.6 accordingly. Our treatment of the data here is based on the analysis of Royle et al. (2011c).

To carry-out an analysis of these data, we require the matrix of trap coordinates and the encounter history data. We store data in an the “scr flat format” (see sec. 6.4.1 above), an efficient file format which is easily manipulated and also used as the input file format in our custom **R** script (ch. xxx) and **SPACECAP** (Gopalaswamy, 2012). To illustrate this format, the wolverine data are available as an encounter data **R** object named “**wcaps**” which has 3 columns and 115 rows, each representing a unique encounter event including the trap identity, the individual identity and the sample occasion index (**sample**). The first 10 rows of this matrix are as follows:

```
> wcaps
      trapid individual sample
```

3414	[1,]	1	2	127
3415	[2,]	1	2	128
3416	[3,]	1	2	129
3417	[4,]	1	18	130
3418	[5,]	2	3	106
3419	[6,]	2	18	104
3420	[7,]	5	5	73
3421	[8,]	5	5	89
3422	[9,]	6	18	117
3423	[10,]	6	18	118

3424 This “encounter data file” contains 1 row for each unique individual/trap en-
 3425 counter, and 3 variables (columns): `trapid` is an integer that runs from `1:ntraps`,
 3426 individual runs from `1:nind` and sample runs from `1:nperiods`. Often (as the case
 3427 here) “sample” will correspond to daily sample intervals. The variable `trapid` will
 3428 have to correspond to the row of a matrix containing the trap coordinates - a file
 3429 named `traplocs.csv` available in the **R** package `scrbook`.

3430 Note that these data do not represent a completely informative summary of the
 3431 data. For example, if no individuals were captured in a certain trap or during a
 3432 certain period, then this compact data format will have no record. Thus we will
 3433 need to know `ntraps` and `nperiods` when reformatting this SCR data format into a
 3434 2-d encounter frequency matrix or 3-d array. In addition, the encounter data file
 3435 does not provide information about which periods each trap was operated. This
 3436 additional information is also necessary as the trap-specific sample sizes must be
 3437 passed to **BUGS** as data. We provide this information in a 2nd data file - which
 3438 we call the “trap deployment” file (described below).

3439 The “encounter data file” `wcaps.csv` exists in the **R** package `scrbook` as a .csv
 3440 file that people can read into **R** and do some basic summary statistics on. For our
 3441 purposes we need to convert these data into the “individual x trap” array of binary
 3442 encounter frequencies, although more general models might require an encounter-
 3443 history formulation of the model which requires a full 3-d array. To obtain our `nind`
 3444 x `ntrap` encounter frequency matrix, we do this the hard way by first converting the
 3445 encounter data file into a 3-d array and then summarize to trap totals. We have a
 3446 handy function `SCR23darray.fn` which takes the compact encounter data file with
 3447 optional arguments `ntraps` and `nperiods`, and converts it to a 3-d array, and then
 3448 we use the **R** function `apply` to summarize over the “sample” period dimension (by
 3449 convention here, this is the 2nd dimension):

```

3450 SCR23darray.fn <- function(caps,ntraps=NULL,nperiods=NULL){
3451   nind<-max(caps[,2])
3452   if(is.null(ntraps)) ntraps<-max(caps[,1])
3453   if(is.null(nperiods)) nperiods<- max(caps[,3])
3454
3455   y<-array(0,c(nind,nperiods,ntraps))
3456   tmp<-cbind(caps[,2],caps[,3],caps[,1])
3457   y[tmp]<-1

```

```

3458 y
3459 }
3460
3461 # for the wolverine data do this:
3462
3463 Y3d <-SCR23darray.fn(wcaps,ntraps=37,nperiods=165)
3464 y <- apply(y3d,c(1,3),sum)

```

If `ntraps` and `nperiods` are not specified then they are assumed to be equal to the maximum value provided in the encounter data file. The 3-d array is necessary to fit certain types of models (e.g., behavioral response) and this is why we sometimes will require this maximally informative 3-d data format.

The other data file that we must have is the “trap deployment” file (henceforth “traps file”) which provides the additional information not contained in the encounter data file. The traps file has `nperiods + 3` columns. The first column is assumed to be a trap identifier, columns 2 and 3 are the easting and northing coordinates (assumed to be in a Euclidean coordinate system), and columns 4 to `(nperiods + 3)` are binary indicators of whether each trap was operational in each time period. The first 5 rows (out of 37) and 10 columns (out of 168) of the traps file for the wolverine data (“`wtraps.csv`” in the **R** package `scrbook` are:

```

3477      Trap Easting Northing 1 2 3 4 5 6 7 <- column names
3478      1   39040   19216 0 0 0 0 0 0 0
3479      2   41324   19772 1 1 1 1 1 1 1
3480      3   44957   12985 0 0 0 0 0 0 0
3481      4   41151   23220 0 0 0 0 0 0 0
3482      5   44240   17198 0 0 0 0 0 0 0

```

This tells us that trap 2 was operated in periods 1-7 but the other traps were not operational during those periods. To extract the relevant information to fit the model in **WinBUGS** we do this:

```

3486 traps<- read.csv("wtraps.csv")
3487 traplocs<- traps[,2:3]
3488 K<- apply(traps[,4:ncol(traps)],1,sum)

```

This results in a matrix `traplocs` which contains the coordinates of each trap and a vector `K` containing the number of days that each trap was operational. We now have all the information required to fit a basic SCR model in **WinBUGS**.

Summarizing these data files for the wolverine study, we see that 21 unique individuals were captured a total of 115 times. Most individuals were captured 1-6 times, with 4, 1, 4, 3, 1, and 2 individuals captured 1-6 times, respectively. In addition, 1 individual was captured each 8 and 14 times and 2 individuals each were captured 10 and 13 times. The number of unique traps that captured a particular individual ranged from 1-6, with 5, 10, 3, 1, 1, and 1 individual captured in each of 1-6 traps, respectively, for a total of 50 unique wolverine-trap encounters. These

numbers might be hard to get your mind around whereas some tabular summary is often more convenient. For that it seems natural to tabulate individuals by trap and total encounter frequencies. The spatial information in SCR data is based on multi-trap captures, and so, it is informative to understand how many unique traps each individual is captured in. At the same, it is useful to understand how many total captures we have of each individual because this is, in an intuitive sense, the effective sample size. So, we reproduce Table 1 from Royle et al. (2011c) which shows the trap and total encounter frequencies:

Table 6.1. Individual frequencies of capture for wolverines captured in camera traps in South-east Alaska in 2008. Rows index unique trap frequencies and columns represent total number of captures (e.g., we captured 4 individuals 1 time, necessarily in only 1 trap; we captured 3 individuals 3 times but in 2 different traps)

	No. of captures									
No. of traps	1	2	3	4	5	6	8	10	13	14
1	4	1	0	0	0	0	0	0	0	0
2	0	0	3	3	0	2	1	2	0	0
3	0	0	1	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	0

6.7.1 Fitting the model in WinBUGS

For illustrative purposes here we fit the simplest SCR model with the half-normal distance function although we revisit these data with more complex models in later chapters. The model is summarized by the following 3 components:

- (1) $y_{ij} | \mathbf{s}_i \sim \text{Bin}(K, z_i p_{ij})$
- (2) $p_{ij} = p_0 \exp(-\theta \|\mathbf{s}_i - x_j\|^2)$
- (3) $\mathbf{s}_i \sim \text{Unif}(\mathcal{S})$
- (4) $z_i \sim \text{Bern}(\psi)$

We assume customary flat priors on the structural (hyper-) parameters of the model, $\alpha_0 = \text{logit}(p_0)$, θ and ψ . It remains to define the state-space \mathcal{S} . For this, we nested the trap array (Fig. 6.3) in a rectangular state-space extending 20 km beyond the traps in each cardinal direction. We also considered larger state-spaces up to 50 km to evaluate that choice. The buffer of the state space should be larger enough so that individuals beyond the state-space boundary are not likely to be encountered. Thus some knowledge of typical space usage patterns of the species is useful. The coordinate system was scaled so that a unit distance was equal to 10km, producing a rectangular state-space of dimension 9.88x10.5 units ($\text{area} = 10374 \text{km}^2$) within which the trap array was nested. As a general rule, we recommend scaling the state-space so that it is defined near the origin $(x, y) = (0, 0)$. While the scaling of the

coordinate system is theoretically irrelevant, a poorly scaled coordinate system can produce Markov chains that mix poorly. We fitted this model in **WinBUGS** using data augmentation with $M = 300$ potential individuals, using 3 Markov chains each of 12000 total iterations, discarding the first 2000 as burn-in. [R commands for reading in the data and executing the analysis are as follows:

provide those commands here

The output follows (note, we have a parameter “sigma” which we discuss shortly):

```

3533 Buffer = 10 km
3534 > print(out1$out,digits=2)
3535 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3536 3 chains, each with 12000 iterations (first 2000 discarded)
3537 n.sims = 30000 iterations saved
3538      mean      sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
3539 psi         0.11  0.02   0.07   0.10   0.11   0.13   0.17    1  2400
3540 sigma       1.79  0.29   1.31   1.58   1.75   1.97   2.46    1   600
3541 p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04    1 13000
3542 N          33.02  4.99  25.00  29.00  32.00  36.00  44.00    1  1600
3543 D           4.93  0.75   3.73   4.33   4.78   5.38   6.57    1  1600
3544 beta        0.17  0.05   0.08   0.13   0.16   0.20   0.29    1   600
3545 deviance 441.97 11.49 421.50 434.00 441.20 449.20 466.30    1  6600
3546
3547
3548 Buffer = 20 km
3549 > print(out2$out,digits=2)
3550 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3551 3 chains, each with 12000 iterations (first 2000 discarded)
3552 n.sims = 30000 iterations saved
3553      mean      sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
3554 psi         0.16  0.04   0.10   0.13   0.16   0.18   0.24    1  4200
3555 sigma       1.78  0.32   1.29   1.55   1.73   1.94   2.56    1 20000
3556 p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04    1  3000
3557 N          47.40  9.19  32.00  41.00  46.00  53.00  68.00    1  5900
3558 D           4.57  0.89   3.08   3.95   4.43   5.11   6.55    1  5900
3559 beta        0.17  0.06   0.08   0.13   0.17   0.21   0.30    1 20000
3560 deviance 444.36 11.84 423.60 436.00 443.60 451.80 469.70    1  1800
3561
3562 Buffer = 25 km
3563 > print(out3$out,digits=2)
3564 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3565 3 chains, each with 12000 iterations (first 2000 discarded)
3566 n.sims = 30000 iterations saved
3567      mean      sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
3568 psi         0.19  0.04   0.11   0.16   0.19   0.22   0.29  1.00   790
3569 sigma       1.80  0.34   1.30   1.56   1.75   1.98   2.59  1.01   400

```

```

3570 p0          0.03  0.00  0.02  0.03  0.03  0.03  0.04  1.00  2800
3571 N          56.66 11.47 37.00 48.00 56.00 64.00 82.00 1.00  570
3572 D           4.53  0.92  2.96  3.84  4.48  5.11  6.55  1.00  570
3573 beta        0.17  0.06  0.07  0.13  0.16  0.20  0.30  1.01  400
3574 deviance 444.75 11.87 423.60 436.40 444.00 452.30 469.80 1.00 24000
3575
3576 Buffer = 30 km
3577 > print(out4$out,digits=2)
3578 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3579 3 chains, each with 12000 iterations (first 2000 discarded)
3580 n.sims = 30000 iterations saved
3581      mean      sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
3582 psi         0.23  0.05  0.14  0.19  0.22  0.26  0.34  1.00  1500
3583 sigma       1.79  0.34  1.29  1.55  1.73  1.97  2.58  1.01  560
3584 p0          0.03  0.00  0.02  0.03  0.03  0.03  0.04  1.00 30000
3585 N          67.39 14.12 43.00 57.00 66.00 76.00 98.00 1.00  1200
3586 D           4.54  0.95  2.90  3.84  4.44  5.12  6.60  1.00  1200
3587 beta        0.17  0.06  0.07  0.13  0.17  0.21  0.30  1.01  560
3588 deviance 444.58 11.83 423.60 436.40 443.80 452.20 469.90 1.00 4700
3589
3590 Buffer = 40 km (need to add this)
3591
3592
3593
3594 Buffer = 45 km
3595 > print(out7$out,digits=2)
3596 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3597 3 chains, each with 12000 iterations (first 2000 discarded)
3598 n.sims = 30000 iterations saved
3599      mean      sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
3600 psi         0.36  0.08  0.21  0.30  0.35  0.41  0.53  1  5000
3601 sigma       1.78  0.34  1.29  1.55  1.72  1.95  2.60  1  850
3602 p0          0.03  0.00  0.02  0.03  0.03  0.03  0.04  1  3600
3603 N          106.57 23.34 67.00 90.00 104.00 121.00 157.00 1  3400
3604 D           4.62  1.01  2.90  3.90  4.51  5.25  6.81  1  3400
3605 beta        0.17  0.06  0.07  0.13  0.17  0.21  0.30  1  850
3606 deviance 444.80 11.84 423.60 436.40 444.10 452.30 470.00 1 30000
3607
3608 Buffer = 50 km
3609 > print(out8$out,digits=2)
3610 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
3611 3 chains, each with 12000 iterations (first 2000 discarded)
3612 n.sims = 30000 iterations saved
3613      mean      sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
3614 psi         0.40  0.09  0.23  0.33  0.39  0.45  0.60  1.01 1300
3615 sigma       1.82  0.48  1.30  1.56  1.74  1.97  2.68  1.05  200

```

3616	p0	0.03	0.00	0.02	0.03	0.03	0.03	0.04	1.00	5800
3617	N	118.47	26.81	71.00	100.00	117.00	135.00	176.00	1.01	1200
3618	D	4.52	1.02	2.71	3.82	4.46	5.15	6.72	1.01	1200
3619	beta	0.17	0.06	0.07	0.13	0.17	0.21	0.30	1.05	200
3620	deviance	444.84	11.90	423.90	436.50	444.10	452.20	470.30	1.00	500

3621 We see that the estimated density is roughly consistent as we increase the state-
 3622 space buffer from 20 to 50 *km*. We do note that the data augmentation parameter
 3623 ψ (and, correspondingly, N) increase with the size of the state space in accordance
 3624 with the deterministic relationship $N = D * A$. However, density is constant more
 3625 or less as we increase the size of the state-space beyond a certain point. For the 10
 3626 *km* state-space buffer, we see a noticeable effect on the posterior distribution of D .
 3627 This is not a bug but rather a feature. As we noted above, the state-space is part
 3628 of the model.

3629 One thing we haven't talked about yet is that we can calibrate the desired size
 3630 of the state-space by looking at the estimated home range radius of the species. For
 3631 some models it is possible to convert the parameter θ directly into the home range
 3632 radius (section XXX XYZ). For the half-normal model we interpret the half-normal
 3633 scale parameter σ which is related to θ by $\theta = 1/(2\sigma^2)$ as the radius of a bivariate
 3634 normal movement model.

3635 6.7.2 Conclusion of Analysis

3636 Our point estimate of wolverine density from this study of approximately 4.5 indi-
 3637 viduals/1000 *km*² and a 95% posterior interval is around [2.7, 6.3]. Density is esti-
 3638 mated imprecisely which might not be surprising given the low sample size ($n = 21$
 3639 individuals!). This seems to be a basic feature of carnivore studies although it
 3640 should not (in our view) preclude the study of their populations nor attempts to
 3641 estimate density or vital rates.

3642 It is worth thinking about this model, and these estimates, computed under a
 3643 rectangular state space roughly centered over the trapping array (Fig. 6.3). Does it
 3644 make sense to define the state-space to include, for example, ocean? What are the
 3645 possible consequences of this? What can we do about it? There's no reason at all
 3646 that the state space has to be a regular polygon – we defined it as such here strictly
 3647 for convenience and for ease of implementation in **WinBUGS** where it enables us
 3648 to specify the prior for the activity centers as uniform priors for each coordinate.
 3649 While it would be possible to define a more realistic state-space using some general
 3650 polygon, it might take some effort to implement that in the **BUGS** language (see
 3651 chapter XYZXYZ² for example of a simple case). Alternatively, we recommend
 3652 using a discrete representation of the state-space – i.e., approximate \mathcal{S} by a grid of
 3653 G points. We discuss this in the following section.

²raccoon example or something?

6.8 CONSTRUCTING DENSITY MAPS

One of the most useful aspects of SCR models is that they are parameterized in terms of individual locations - i.e., *where* each individual lives – and, thus, we can compute many useful or interesting summaries of the activity centers. For example, we can make a spatial density plot by tallying up the number of activity centers \mathbf{s}_i in boxes of arbitrary size and then producing a nice multi-color spatial plot of those which, we find, increases the acceptance probability of your manuscripts by a substantial amount. We discussed in chapter 2 the idea of estimating derived parameters from MCMC output. In SCR models, there are many derived parameters that are functions of the latent point locations $(\mathbf{s}_1, \dots, \mathbf{s}_N)$. In the present context, the number of individuals living in any well-defined polygon is a derived parameter. Specifically, let $B(x)$ indicate a box centered at x then

$$N(x) = \sum_i I(\mathbf{s}_i \in B(x))$$

is the population size of box $B(x)$, and $D(x) = N(x)/|B(x)|$ is the local density. These are just “derived parameters” (see chapter 2) which are estimated from MCMC output using the appropriate Monte Carlo average. One thing to be careful about, in the context of models in which N is unknown, is that, for each MCMC iteration m , we only tabulate those activity centers which correspond to individuals in the sampled population. i.e., for which the data augmentation variable $z_i = 1$. In this case, we take all of the output for MCMC iterations $m = 1, 2, \dots, \text{niter}$ and compute this summary:

$$N(x, m) = \sum_{z_{i,m}=1} I(s_{i,m} \in B(x))$$

Thus, $N(x, 1), N(x, 2), \dots$, is the Markov chain for parameter $N(x)$. In what follows we will provide a set of **R** commands for doing this calculations and making a basic image plot from the MCMC output.

Step 1: Define the center points of each box, $B(x)$, or point at which local density will be estimated:

```
xg<-seq(Xl,Xu,,50)
yg<-seq(Yl,Yu,,50)
```

Step 2: Extract the MCMC histories for the activity centers and the data augmentation variables. Note that these are each $N \times \text{niter}$ matrices:

```
Sxout<-out$sims.list$s[,1]
Syout<-out$sims.list$s[,2]
z<-out$sims.list$z
```

3685 **Step 3:** We associate each coordinate with the proper box using the **R** command
 3686 `cut()`. Note that we keep only the activity centers for which $z = 1$ (i.e., individuals
 3687 that belong to the population of size N):

```
3688 Sxout<-cut(Sxout[z==1],breaks=xg,include.lowest=TRUE)
3689 Syout<-cut(Syout[z==1],breaks=yg,include.lowest=TRUE)
```

3690 **Step 4:** Use the `table()` command to tally up how many activity centers are in
 3691 each $B(x)$:

```
3692 Dn<-table(Sxout,Syout)
```

3693 **Step 5:** Use the `image()` command to display the resulting matrix.

```
3694 image(xg,yg,Dn/nrow(z),col=terrain.colors(10))
```

3695 Praise the Lord! This map is somewhat useful or at least it looks pretty and will
 3696 facilitate the publication of your papers.

3697 It is worth emphasizing here that density maps will not usually appear uniform
 3698 despite that we have assumed that activity centers are uniformly distributed. This is
 3699 because the observed encounters of individuals provide direct information about the
 3700 location of the $i = 1, 2, \dots, n$ activity centers and thus their “estimated” locations
 3701 will be affected by the observations. In a limiting sense, were we to sample space
 3702 intensely enough, every individual would be captured a number of times and we
 3703 would have considerable information about all N point locations. Consequently,
 3704 the uniform prior would have almost no influence at all on the estimated density
 3705 surface in this limiting situation. Thus, in practice, the influence of the uniformity
 3706 assumption increases as the fraction of the population encountered decreases.

3707 **On the non-intuitiveness of `image()`** – the **R** function `image()` might not
 3708 be very intuitive to some – it plots $M[1, 1]$ in the lower left corner. If you want $M[]$
 3709 to be plotted “as you look at it” then $M[1, 1]$ should be in the upper left corner.
 3710 We have a function `rot()` which does that. If you do `image(rot(M))` then it puts
 3711 it on the monitor as if it was a map you were looking at. You can always specify
 3712 the x and y – labels explicitly as we did above.

3713 **Spatial dot plots** – Now here is a cruder version based on the “spatial
 3714 dot map” function `spatial.plot`. The useful functions in **R** are `image()` and
 3715 `image.scale()` which is a function we grabbed off the web somewhere. Use of
 3716 this function requires arguments of point locations and the resulting value to be
 3717 displayed. The function is defined and applied as follows:

```
3718 spatial.plot<- function(x,y){
3719   nc<-as.numeric(cut(y,20))
3720   plot(x,pch=" ")
3721   points(x,pch=20,col=topo.colors(20)[nc],cex=2)
3722   image.scale(y,col=topo.colors(20))
```

```

3723 }
3724 # To execute the function do this:
3725 spatial.plot(cbind(xg,yg), Dn/nrow(z))

```

3726 6.8.1 Example: Wolverine density map.

3727 We used the posterior output from the wolverine model fitted previous to compute
 3728 a relatively coarse version of a density map, using a 10×10 grid (Fig. 6.4) and
 3729 using a 30×30 grid (Fig. 6.5)³. In these figures density is expressed in units of
 3730 individuals per 1000 km^2 , while the area of the pixels is about 1037 km^2 and 115
 3731 km^2 , respectively. That calculation is based on⁴:

```

3732 > total.area<- (Yu-Yl)*(Xu-Xl)*1000
3733 > total.area/(10*10)
3734 [1] 1037.427
3735 > total.area/(30*30)
3736 [1] 115.2697

```

3737 A couple of things are worth noting: First is that as we move away from “where
 3738 the data live” - away from the trap array - we see that the density approaches
 3739 the mean density. This is a property of the estimator as long as the “detection
 3740 function” decreases sufficiently rapidly as a function of distance. Relatedly, it is
 3741 also a property of statistical smoothers such as splines, kernel smoothers, and re-
 3742 gression smoothers - predictions tend toward the global mean as the influence of
 3743 data diminishes. Another way to think of it is that it is a consequence of the prior
 3744 - which imposes uniformity, and as you get far away from the data, the predictions
 3745 tend to the prior. The other thing to note about this map is that density is not
 3746 0 over water (although the coastline is not shown). This might be perplexing to
 3747 some who are fairly certain that wolverines do not like water. However, there is
 3748 nothing about the model that recognizes water from non-water and so the model
 3749 predicts over water *as if* it were habitat similar to that within which the array is
 3750 nested. But, all of this is ok as far as estimating density goes and, furthermore, we
 3751 can compute valid estimates of N over any well-defined region which presumably
 3752 wouldn’t include water if we so choose.

6.9 DISCRETE STATE-SPACE

3753 The SCR model developed previously in this chapter assumes that individual activ-
 3754 ity centers are distributed uniformly over the prescribed state-space. Clearly this
 3755 will not always be a reasonable assumption. In chapter 15 we talk about developing

³Note: Not sure if we should use quantiles for color to make equal area slices. ??? Also should we use the same scale?

⁴This is wrong and needs fixed. Move decimal one place over. i.e., 100 instead of 1000.

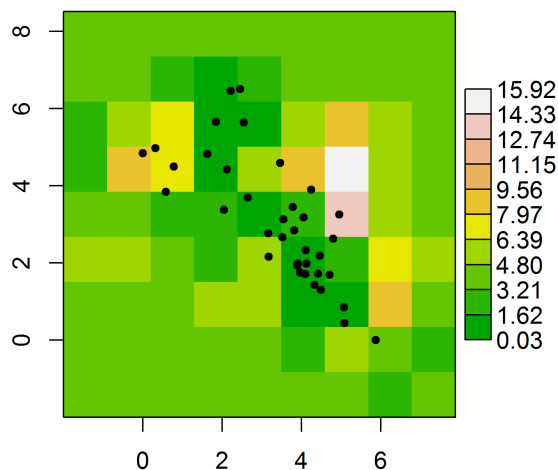


Figure 6.4. Needs a caption

models that allow explicitly for non-uniformity of the activity centers by modeling covariate effects on density. A simpler method of affecting the distribution of activity centers, which we address here, is to modify the shape of the state-space explicitly. For example, we might be able to classify the state-space into distinct blocks of habitat and non-habitat. In that case we can remove the non-habitat from the state-space and assume uniformity of the activity centers over the remaining portions judged to be suitable habitat. There are two ways to approach this: We can use a regular grid of points to represent the state-space, i.e., by the set of coordinates $\mathbf{s}_1, \dots, \mathbf{s}_G$, and assign a equal probabilities to each possible value, or we can retain the continuous formulation of the state-space but use basic polygon operations to induce constraints on the state-space We focus here on the formulation of our basic SCR model in terms of a discrete state-space but later on (chapter 10 and also Appendix XYZ) we demonstrate the latter approach based on using polygon operations to define an irregular state-space.

Use of a discrete state-space can be computationally expensive in **WinBUGS**. That said, it isn't too difficult to do the MCMC calculations in **R** which we discuss briefly in chapter 10. The **R** package **SPACECAP** (Gopalaswamy et al., 2011) arose from the **R** implementation developed for the application in Royle et al. (2009). As we will see in chapter 9, we must prescribe the state-space by a discrete mesh of points in order to do integrated likelihood and so if we are using a discrete

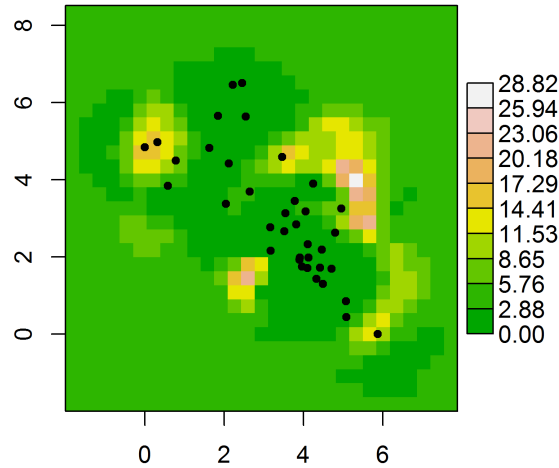


Figure 6.5. Needs a caption

state-space this can be accommodated directly in our code for obtaining MLEs.

While clipping out non-habitat seems like a good idea, its not obvious that we accomplish any biologically reasonable objective by doing so. We might prefer to do it when non-habitat represents a clear-cut restriction on the state-space such as a reserve boundary or a lake, ocean or river. It makes sense in those situations. Unfortunately, having the capability to do this also causes people to start defining “habitat” vs. “non-habitat” based on their understanding of the system whereas it can’t be known whether the animal being studied has the same understanding. Moreover, differentiating of the landscape by habitat or habitat quality probably affects the geometry and morphology of home ranges much more than the plausible locations of activity centers. That is, a home range centroid could, in actual fact, occur in a walmart parking lot if there is pretty good habitat around walmart, so there is probably no sense to cut out the walmart lot and preclude it as the location for an activity center. It would generally be better to include some definition of habitat quality in the model for the detection probability (see chapter XYZ).

6.9.1 Evaluation of Coarseness of Discrete Approximation

The coarseness of the state-space should not really have much of an effect on estimates if the grain is sufficiently fine relative to typical animal home range sizes.

Why is this? We have two analogies that can help us understand this. First is the relationship to Model M_h . As noted in section 6.3.4 above, we can think about SCR models as a type of finite mixture (Norris III and Pollock, 1996; Pledger, 2000) where we are fortunate to be able to obtain direct information about which “group” individuals belong to (group being location of activity center). In the standard finite mixture models we typically find that only 1 or a very small number of groups (e.g., 2 or 3 at the most) can explain really high levels of heterogeneity and are adequate for most data sets of small to moderate sample sizes. We therefore expect a similar effect in SCR models when we discretize the state-space. We can also think about discretizing the state-space as being related to numerical integration where we find (see chapter 9) that we don’t need a very fine grid of support points to evaluate the integral to a reasonable level of accuracy. We demonstrate this here by reanalyzing simulated data using a state-space defined by a different numbers of support points. We provide an R script called `simSCR0discrete.fn` in the R package `scrbook`. We note that for this comparison we generated the actual activity centers as a continuous random variable and thus the discrete state-space is, strictly speaking, an approximation to truth. That said, we regard all state-space specifications as approximations to truth because they are all, strictly speaking, models of some unknown truth. Thus the use of any specific discrete state-space is not intrinsically more “wrong” than any specific continuous representation.

We used **JAGS** from the `rjags` function to obtain the results for 6×6 , 9×9 , 12×12 , 15×15 , 20×20 , 25×25 and 30×30 state-space grids. We used 2000 burn, 12000 total iters with 3 chains, therefore a total of 30000 posterior samples. For **WinBUGS** we used 3 chains of 5k total with 1k burnin means 12k total posterior samples. Summary results for these analyses are shown in Table XYZ⁵.

Table XYZ.

			Mean	SD	NaiveSE	Time-seriesSE	runtime
6	N		109.7717	15.98959	0.0923160	0.377737	1239
9	N		114.4621	16.72025	0.0965344	0.468659	1267
12	N		115.4309	17.12403	0.098866	0.464830	1576
15	N		114.7699	17.0242	0.0982894	0.425238	1638
20	N		116.0370	17.10686	0.0987665	0.486867	1647
25	N		116.3228	16.98323	0.0980527	0.465527	1661
30	N		116.4252	17.4078	0.100504	0.533735	1806
WinBUGS							
			Mean	SD	NaiveSE	Time-seriesSE	runtime
6	N		111.67	16.61			2274
9	N		114.23	17.99			4300
12	N		115.98	17.38			7100
15	N		115.38	17.94			13010

⁵Andy to finish later

3835 Note: WinBUGS based on fewer samples too!

3836

3837 To get SE and time-series SE do this:

3838 You can use `as.mcmc.list()` to convert to a coda object. Then use `summary`.

3839 The results in terms of the posterior summaries are, as we expect, very similar
 3840 using **WinBUGS**. However, it was interesting to note that **WinBUGS** runtime is
 3841 much worse (note the number of iterations is lower for **WinBUGS** yet the runtime
 3842 is much longer) and, furthermore, it seems to scale with the size of the discrete
 3843 state-space grid. While that was expected, it was unexpected that the runtime of
 3844 **JAGS** would seem relatively consistent as we increase the grid size. We suspect
 3845 that **WinBUGS** is evaluating the full-conditional for each activity center at all
 3846 G possible values whereas it may be that **JAGS** is evaluating the full-conditional
 3847 only at a subset of values or perhaps using previous calculations more effectively.

3848 While this might suggest that one should always use **JAGS** for this analysis, we
 3849 found in our analysis of the wolverine (next section) that **JAGS** could be extremely
 3850 sensitive to starting values, producing MCMC algorithms that sometimes simply
 3851 did not work.

3852 6.9.2 Analysis of the wolverine camera trapping data

3853 We reanalyzed the wolverine data using discrete state-space grids with points spaced
 3854 by 2, 4 and 8 km (depicted in Fig. ??). These were constructed from the 40 km
 3855 buffered state-space, and deleting the points over water (see Royle et al., 2011c).
 3856 Our interest in doing this was to evaluate the relative influence of grid resolution
 3857 on estimated density because the coarser grids will be more efficient from a compu-
 3858 tational stand-point and so we would prefer to use them, but perhaps not if there
 3859 is a strong influence on estimated density.

3860 **Note:** Results from WinBUGS are given below based on short runs that took
 3861 a long long time. I am rerunning those. I will also show a density map for each
 3862 analysis.

3863 based on 2k burn 3k total and 3 chains = 3k total posterior samples.
 3864 lots of MC error here.

3865

3866 2km

3867 For each parameter, `n.eff` is a crude measure of effective sample size,
 3868 and `Rhat` is the potential scale reduction factor (at convergence, `Rhat=1`).

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
3869 psi	0.28	0.06	0.17	0.24	0.27	0.32	0.41	1.01	230
3870 sigma	0.64	0.05	0.55	0.60	0.64	0.67	0.73	1.02	88
3871 lam0	-3.00	0.16	-3.33	-3.11	-3.00	-2.90	-2.69	1.04	52
3872 p0	0.05	0.01	0.03	0.04	0.05	0.05	0.06	1.04	52
3873 N	82.95	16.26	55.00	72.00	82.00	93.00	119.02	1.01	240

```

3875
3876 4 km
3877 For each parameter, n.eff is a crude measure of effective sample size,
3878 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
3879      mean      sd 2.5% 25% 50% 75% 97.5% Rhat n.eff
3880 psi    0.30 0.06 0.19 0.26 0.29 0.34 0.43 1.01 580
3881 sigma 0.62 0.05 0.54 0.59 0.62 0.65 0.72 1.00 2000
3882 lam0 -3.00 0.16 -3.33 -3.10 -2.99 -2.90 -2.67 1.01 390
3883 p0     0.05 0.01 0.03 0.04 0.05 0.05 0.06 1.01 390
3884 N      88.78 16.76 60.00 77.00 87.00 99.00 125.00 1.01 690
3885
3886 8km
3887 For each parameter, n.eff is a crude measure of effective sample size,
3888 and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
3889      mean      sd 2.5% 25% 50% 75% 97.5% Rhat n.eff
3890 psi    0.27 0.06 0.17 0.23 0.27 0.31 0.40 1.00 1500
3891 sigma 0.69 0.05 0.60 0.65 0.68 0.72 0.80 1.00 3000
3892 lam0 -3.07 0.17 -3.41 -3.20 -3.07 -2.95 -2.74 1.01 210
3893 p0     0.04 0.01 0.03 0.04 0.04 0.05 0.06 1.01 200
3894 N      82.01 15.98 55.00 71.00 80.00 92.00 118.00 1.00 1300

```

3895 We did the analysis in JAGS also. The results are shown below. **Note:** I am
3896 going to run these again but for longer to finalize the results.

```

3897 2km
3898 Iterations = 7001:13000
3899 Thinning interval = 1
3900 Number of chains = 3
3901 Sample size per chain = 6000
3902
3903      Mean      SD Naive SE Time-series SE
3904 N      86.28522 16.950626 1.263e-01 0.4878973
3905 lam0 0.04807 0.007512 5.599e-05 0.0002199
3906 p0 0.04581 0.006820 5.083e-05 0.0001996
3907 psi 0.28904 0.062117 4.630e-04 0.0017481
3908 sigma 0.62769 0.043596 3.249e-04 0.0018724
3909
3910 4km
3911      Mean      SD Naive SE Time-series SE
3912 N      85.53139 16.998966 1.267e-01 0.5181297
3913 lam0 0.04636 0.007542 5.621e-05 0.0002382
3914 p0 0.04425 0.006867 5.118e-05 0.0002172
3915 psi 0.28650 0.061922 4.615e-04 0.0018276
3916 sigma 0.64281 0.048321 3.602e-04 0.0022911
3917
3918 8km
3919      Mean      SD Naive SE Time-series SE

```


3920	N	83.97039	16.508146	1.230e-01	0.4548782
3921	lam0	0.04519	0.006919	5.157e-05	0.0001738
3922	p0	0.04319	0.006319	4.710e-05	0.0001589
3923	psi	0.28146	0.060653	4.521e-04	0.0016555
3924	sigma	0.66956	0.040989	3.055e-04	0.0015070

3925 6.9.3 SCR models as multi-state models

3926 While we invoke a discrete state-space artificially, by gridding the underlying con-
 3927 tinuous state-space, sometimes the state-space is more naturally discrete. Consider
 3928 a situation in which discrete patches of habitat are searched using some method
 3929 and it might be convenient (or occur inadvertently) to associate samples to the
 3930 patch level instead of recording observation locations. In this case we might use a
 3931 model $\mathbf{s}_i \sim \text{dcat}(\text{probs}[])$ where $\text{probs}[]$ are the probabilities that an individual in-
 3932 habits a particular patch. We consider such a case study in chapter XXPoissonXXX
 3933 from Mollet et al. (2012) who obtained a population size estimate of a large grouse
 3934 species known as the capracaille. Forest patches were searched for scat which was
 3935 identified to individual by DNA analysis. Even when space is *not* naturally discrete,
 3936 measurements are often made at a fairly coarse grain (e.g., meters or tens of meters
 3937 along a stream), or associated with spatial quadrats for scat searches and therefore
 3938 the state-space may be effectively discrete in many situations.

3939 This discrete formulation of SCR models suggests that SCR models are related
 3940 to ordinary multi-state models (Kery and Schaub, 2011, ch. 9) which are also
 3941 parameterized in terms of a discrete state variable which is often defined as a
 3942 spatially-indexed state related either to location of capture or breeding location.
 3943 While many multi-state models exist in which the state variable is not related to
 3944 space, multi-state models have been extremely useful in development models of
 3945 movements among geographic states and indeed this type of problem motivated
 3946 their early developments by Arnason (1973, 1974) and Hestbeck (1991). We pursue
 3947 this connection a little bit more in chapter XXX XYZ.

6.10 SUMMARY AND OUTLOOK

3948 A point we tried to emphasize in this chapter is that the basic SCR model is not
 3949 much more than an ordinary capture-recapture model for closed populations – it
 3950 is simply that model but augmented with a set of “individual effects”, \mathbf{s}_i , which
 3951 relate encounter probability to some sense of individual location. SCR models are
 3952 therefore a type of individual covariate model (as introduced in chapter 11 – but
 3953 with imperfect information about the individual covariate. In other words, they
 3954 are GLMM type models when N is known or, when N is unknown, they are zero-
 3955 inflated GLMMs (see Royle (2006)). Another class of capture-recapture models
 3956 that SCR models are closely related to is so-called “Model M_h .” The effect of
 3957 introducing a spatial location for individuals is that it induces heterogeneity in

detection probability, as in Model M_h . However, unlike Model M_h , we obtain some information about the individual effect which is completely latent in Model M_h . If the state-space of the random effect \mathbf{s} is discrete then the SCR model resembles more closely the finite-mixture class of heterogeneity models (Norris III and Pollock, 1996) which parameterizes heterogeneity by assuming that individuals belong to discrete classes or groups (e.g., high, medium, low). In the context of SCR models we obtain some information about the “group membership” in the locations where individuals are captured. Given the direct relationship of SCR models with so many standard classes of models, we find that they are really quite easy to analyze using standard MCMC methods encased in black boxes such as **WinBUGS** or **JAGS** and possibly other packages. They are also easy to analyze using classical likelihood methods, which we address in chapter 9.

Formal consideration of the collection of individual locations $(\mathbf{s}_1, \dots, \mathbf{s}_N)$ in the model is fundamental to all of the models considered in this book. In statistical terminology, we think of the collection of points $\{\mathbf{s}_i\}$ as a realization of a point process and part of the promise, and ongoing challenge, of SCR models is to develop models that reflect interesting biological processes, for example interactions among points or temporal dynamics in point locations. Here we considered the simplest possible point process model - the points are independent and uniformly (“randomly”) distributed over space. Despite the simplicity of this assumption, it should suffice in many applications of SCR models although we do address generalizations of this model in later chapters. Moreover, even though the *prior* distribution on the point locations is uniform, the realized pattern may deviate markedly from uniformity as the observed encounter data provide information to impart deviations from uniformity. Thus, the estimated density map will typically appear distinctly non-uniform. As a general rule, information in the data will govern estimates of individual point locations so even fairly complex patterns of non-independence or non-uniformity will appear in the data. That is, we find in applications of the basic SCR model that this simple *a priori* model can effectively reflect or adapt to complex realizations of the underlying point process. For example, if individuals are highly territorial then the data should indicate this in the form of individuals not being encountered in the same trap - the resulting posterior distribution of point locations should therefore reflect non-independence. Obviously the complexity of posterior estimates of the point pattern will depend on the quantity of data, both number of individuals and captures per individual. Because the point process is such an integral component of SCR models, the state-space of the point process plays an important role in developing SCR models. As we tried to emphasize in this chapter, the choice of the state-space is part of the model. It can have an influence on parameter estimates and other inferences such as model selection (see chapter 12). We emphasize however that this is not an arbitrary decision like “buffering” because the model induces an explicit interpretation of parameters and statistical effect on estimators.

We showed how to conduct inference about the underlying point process includ-

ing calculation of density maps from posterior output. We can do other things we normally do with spatial point processes such as compute “K-functions” and test for “complete spatial randomness” (CSR) which we develop in chapter 12. Modifying and applying point process methods to SCR problems seems to us to be a fruitful area of research.

An obvious question that might be floating around in your mind is why should we ever go through all of this trouble when we could just use **MARK** or **CAPTURE** to get an estimate of N and apply 1/2 MMDM methods? The main reason is that these conventional methods are predicated on models that represent explicit misspecifications of both the observation and ecological process - they are wrong! Not just wrong, because of course all models are wrong, but they’re not even *plausible* models! Thus while we might be able to show adequate fit or whatever, we think as a conceptual and philosophical model one should not be using models that are not even plausible data-generating models – even if the plausible ones don’t fit! Perhaps more charitably, these ordinary non-spatial models are models of the wrong system. They do not account for trap identity. They don’t account for spatial organization or “clustering” of individual encounters in space. And, “density” is not a parameter of those models because density has no meaning absent an explicit representation of space. If we do define space explicitly, e.g., as a buffered minimum convex hull, then the normal models (M_0 , M_h , etc..) assume that individual capture-probability is not related to space, no matter how we define the buffer. Conversely, the SCR model is a model for trap-specific encounter data - how individuals are organized in space and interact with traps. SCR models provide a coherent framework for inference about density or population size and also, because of the formality of their derivation, can be extended and generalized to a large variety of different situations, as we demonstrate in subsequent chapters.

In the next few chapters we continue to work with this basic SCR design and model but consider some important extensions of the basic model. For example, we consider extensions to include covariates that vary by individual, trap, or over time (chapter 13), spatial covariates on density (chapter 15), open populations (chapter 16), model assessment and selection (chapter 12) and other topics. We also consider technical details of Bayesian (chapter 10) and maximum likelihood (chapter 9) estimation so that the interested reader can develop or extend their own methods to suit their needs.

7

4035

4036

4037

OTHER OBSERVATION MODELS

8

4038
4039
4040

MAXIMUM LIKELIHOOD ESTIMATION

LIKELIHOOD ANALYSIS OF SCR MODELS

In this book we mainly focus on Bayesian analysis of spatial capture-recapture models. And, in the previous chapters we learned how to fit some basic spatial capture-recapture models using a Bayesian formulation of the models analyzed in BUGS engines including **WinBUGS** and **JAGS**. Despite our focus on Bayesian analysis, it is instructive to develop the basic conceptual and methodological ideas behind classical analysis based on likelihood methods and frequentist inference. In fact, simple SCR models can be analyzed fairly easily using such methods. This has been the approach taken by Borchers and Efford (2008); Dawson and Efford (2009) and related papers.

This chapter provides some conceptual and technical footing for likelihood-based analysis of spatial capture-recapture models. We recognized earlier (chapt. 4) that SCR models are versions of binomial (or other) GLMs, but with random effects i.e., GLMMs. These models are routinely analyzed by likelihood methods. In particular, likelihood analysis is based on the integrated likelihood in which the random effects are removed by integration from the likelihood. In SCR models, the random effect, \mathbf{s} , i.e., the 2-dimensional coordinate, is a bivariate random effect. In this chapter, we show that it is straightforward to compute the maximum likelihood estimates (MLE) for SCR models by integrated likelihood. We develop the MLE framework using **R**, and we also provide a basic introduction to an **R** package **secr** (Efford, 2011) which is based on the stand-alone package **DENSITY** (Efford et al., 2004). To set the context we analyze the SCR model here when N is known because, in that case, it is precisely a GLMM and does not pose any difficulty at all. We generalize the model to allow for unknown N using both conventional ideas based on the “joint likelihood” (e.g., Borchers et al., 2002) and also using a formulation based on data augmentation. We consider likelihood analysis of SCR models in the context of the wolverine camera trapping study (Magoun et al., 2011) we analyzed

in previous chapters to compare/contrast the results.

9.1 LIKELIHOOD ANALYSIS

We noted in chapter 4 that, with N known, the basic SCR model is a type of binomial regression with a random effect. For such models we can easily obtain maximum likelihood estimators of model parameters based on integrated likelihood. The integrated likelihood is based on the marginal distribution of the data y in which the random effects are removed by integration. Conceptually, our model is a specification of the conditional-on- \mathbf{s} model $[y|\mathbf{s}, \theta]$ and we have a “prior distribution” for \mathbf{s} , say $[s]$, and the marginal distribution of the data y is

$$[y|\theta] = \int_{\mathbf{s}} [y|\mathbf{s}, \theta][\mathbf{s}]d\mathbf{s}.$$

When viewed as a function of θ for purposes of estimation, the marginal distribution $[y|\theta]$ is often referred to as the *integrated likelihood*.

It is worth analyzing the simplest SCR model with known- N in order to understand the underlying mechanics and basic concepts. These are directly relevant to the manner in which many capture-recapture models are classically analyzed, such as model Mh, and individual covariate models (see chapt. 6 from Royle and Dorazio (2008)). To develop integrated likelihood for SCR models, we first identify the conditional likelihood.

The observation model for each encounter observation y_{ij} , specified conditional on \mathbf{s}_i , is

$$y_{ij}|\mathbf{s}_i \sim \text{Bin}(K, p_{\theta}(\mathbf{x}_j, \mathbf{s}_i)) \quad (9.1.1)$$

where we have indicated the dependence of p_{ij} on \mathbf{s} and parameters θ explicitly. For the random effect we have $\mathbf{s}_i \sim \text{Unif}(\mathcal{S})$. The joint distribution of the data for individual i is the product of J such terms (i.e., contributions from each of J traps).

$$[\mathbf{y}_i|\mathbf{s}_i, \theta] = \prod_j \text{Bin}(K, p_{\theta}(\mathbf{x}_j, \mathbf{s}_i))$$

We note that this assumes that encounter of individual i in each trap is independent of encounter in every other trap, conditional on \mathbf{s}_i , this is the fundamental property of SCR0 or “multi-catch” traps.

The so-called “marginal likelihood” is computed by removing \mathbf{s}_i , by integration, from the conditional-on- \mathbf{s} likelihood. That is, we compute:

$$[y|\theta] = \int_{\mathcal{S}} \mathcal{L}(\theta|\mathbf{y}_i|\mathbf{s}_i)g(\mathbf{s}_i)d\mathbf{s}_i$$

here $g(s) = 1/|\mathcal{S}|$.

4098 The joint likelihood for all N individuals, assuming independence of encounters
4099 among individuals, is the product of N such terms:

$$\prod_i f(y[i, j])$$

4100 We emphasize that two independence assumptions are explicit in this development:
4101 independence of trap-specific encounters within individuals and also independence
4102 among individuals. In particular, this would only be valid when individuals are not
4103 physically restrained or removed upon capture, and when traps do not “fill up”.

4104 The key operation for computing the likelihood is solving a 2-dimensional inte-
4105 gration problem. There are some general purpose **R** packages that implement
4106 a number of multi-dimensional integration routines including **adapt** (Genz et al.,
4107 2007) and **R2cuba** (Hahn et al., 2011). In practice, we won’t rely on these extra-
4108 neous **R** packages but instead will use perhaps less efficient methods in which we
4109 replace the integral with a summation over an equal area mesh of points on the
4110 state-space \mathcal{S} and explicitly evaluate the integrand at each point. We invoke the
4111 rectangular rule for integration here¹ in which we evaluate the integrand on a regu-
4112 lar grid of points of equal area and compute the average of the integrand over that
4113 grid of points. Let $u = 1, 2, \dots, nG$ index a grid of nG points, \mathbf{s}_u , where the area
4114 of grid cell u is A . In this case, the integrand, i.e., the marginal probability of y_{ij} ,
4115 is approximated by

$$f(y_{ij}|\theta) = \frac{1}{nG} \sum_{u=1}^{nG} f(\mathbf{y}_i|\mathbf{s}_u) \quad (9.1.2)$$

4116 This is a specific case of the general expression that could be used for approxi-
4117 mating the integral for any arbitrary bivariate distribution $g(u)$. The general case
4118 is

$$[y] = \frac{A}{nG} \sum_u [y|\mathbf{s}_u][\mathbf{s}_u]$$

4119 In the present context note that $[\mathbf{s}] = (1/A)$ and thus the grid-cell area cancels in
4120 the above expression to yield eq. 9.1.2.

4121 Not surprisingly this the same answer we get if \mathbf{S} were inherently discrete, having
4122 nG unique values with equal probabilities $1/nG$, and we apply the Law of Total
4123 Probability directly to compute the marginal probability $[y|\theta]$.

4124 9.1.1 Implementation (simulated data)

4125 Here we will illustrate how to carryout this integration and optimization based on
4126 the integrated likelihood using simulated data (i.e., following that from Chapter 4).
4127 Using **simSCRO.fn** we simulate data for 100 individuals and a 25 trap array layed

¹e.g., http://en.wikipedia.org/wiki/Rectangle_method

4128 out in a 5×5 grid of unit spacing. The specific encounter model is the half-normal
 4129 model. The 100 activity centers were simulated on a state-space defined by a 8×8
 4130 square within which the trap array was centered (thus the trap array is buffered by
 4131 2 units). Therefore, the density of individuals in this system is fixed at 100/64.

4132 In the following set of R commands we generate the data and then harvest the
 4133 required data objects:

```
4134 data<-simSCR0.fn(discard0=FALSE,sd=2013)
4135 y<-data$Y
4136 traplocs<-data$traplocs
4137 nind<-nrow(y)
4138 X<-data$traplocs
4139 J<-nrow(X)
4140 K<-data$K
4141 Xl<-data$xlim[1]
4142 Yl<-data$ylim[1]
4143 Xu<-data$xlim[2]
4144 Yu<-data$ylim[2]
```

4145 Now we need to define the integration grid, say **G**, which we do with the following
 4146 set of **R** commands (here, **delta** is the grid spacing):

```
4147 delta<- .2
4148 xg<-seq(Xl+delta/2,Xu-delta/2,by=delta)
4149 yg<-seq(Yl+delta/2,Yu-delta/2,by=delta)
4150 npix<-length(xg)      # assumes xg and yg same dimension here
4151 area<- (Xu-Xl)*(Yu-Yl)/((npix)*(npix)) # dont need area for anything
4152 G<-cbind(rep(xg,npix),sort(rep(yg,npix)))
4153 nG<-nrow(G)
```

4154 In this case, the integration grid is set up as a grid with spacing $\delta = 0.2$ which
 4155 produces a 40×40 grid of points for evaluating the integrand if the state-space
 4156 buffer is set at 2.

4157 We next create an **R** function that defines the likelihood as a function of the
 4158 data objects **y** and **X** which were created above but, in general, you would read
 4159 these files into **R**, e.g., from a .csv file. In addition to these data objects, we need
 4160 to have defined the various quantities associated with the integration grid **G** and
 4161 **nG**. However, instead of worrying about making all of these objects and keeping
 4162 track of them we just put that code above into the likelihood function and pass δ
 4163 as an additional (optional) argument and a few other things that we need such as
 4164 the boundary of the state-space over which the integration (summation) is being
 4165 done. Here is one reasonably useful variation of a function for estimation based on
 4166 the integrated likelihood:

4167

```

4168 intlik1<-function(parm,y=y,delta=.2,X=traplocs,ssbuffer=2){
4169
4170   Xl<-min(X[,1]) - ssbuffer
4171   Xu<-max(X[,1]) + ssbuffer
4172   Yl<-min(X[,2]) - ssbuffer
4173   Yl<-min(X[,2]) - ssbuffer
4174
4175   xg<-seq(Xl+delta/2,Xu-delta/2,,length=npix)
4176   yg<-seq(Yl+delta/2,Yu-delta/2,,length=npix)
4177   npix<-length(xg)
4178
4179   G<-cbind(rep(xg,npix),sort(rep(yg,npix)))
4180   nG<-nrow(G)
4181   D<- e2dist(X,G)
4182
4183   alpha0<-parm[1]
4184   alpha1<-parm[2]
4185   probcap<- plogis(alpha0)*exp(-alpha1*D*D)
4186   Pm<-matrix(NA,nrow=nrow(probcap),ncol=ncol(probcap))
4187   # all zero encounter histories
4188   n0<-sum(apply(y,1,sum)==0)
4189   # encounter histories with at least 1 detection
4190   ymat<-y[apply(y,1,sum)>0,]
4191   ymat<-rbind(ymat,rep(0,ncol(ymat)))
4192   lik.marg<-rep(NA,nrow(ymat))
4193   for(i in 1:nrow(ymat)){
4194     Pm[1:length(Pm)]<- (dbinom(rep(ymat[i,],nG),K,probcap[1:length(Pm)],log=TRUE))
4195     lik.cond<- exp(colSums(Pm))
4196     lik.marg[i]<- sum( lik.cond*(1/nG))
4197   }
4198   nv<-c(rep(1,length(lik.marg)-1),n0)
4199   -1*( sum(nv*log(lik.marg)) )
4200 }

```

4201 The function accepts as input the encounter history matrix, y , the trap locations,
 4202 X , and the state-space buffer. This allows us to vary the state-space buffer and
 4203 easily evaluate the sensitivity of the MLE to the size of the state-space. Note that
 4204 we have a peculiar handling of the encounter history matrix y . In particular, we
 4205 remove the all-zero encounter histories from the matrix and tack-on a single all-zero
 4206 encounter history as the last row which then gets weighted by the number of such
 4207 encounter histories (n_0). This is a bit long-winded and strictly unnecessary when
 4208 N is known, but we did it this way because the extension to the unknown- N case
 4209 is now transparent (as we demonstrate in the following section). The matrix Pm
 4210 holds the log-likelihood contributions of each encounter frequency for each possible
 4211 state-space location of the individual. The log contributions are summed up and
 4212 the result exponentiated on the next line, producing $lik.cond$, the conditional-on-

s likelihood (Eq. 9.1.1 above). The marginal likelihood (`lik.marg`) sums up the conditional elements weighted by $\Pr(\mathbf{s})$ (formula XXX above). Finally, this function assumes that K , the number of replicates, is constant for each trap. Further, it assumes that the state-space is a square. As an exercise, consider resolving these two issues by generalizing the code.

Here is the **R** command for maximizing the likelihood and saving the results into an object called `frog`. The output is a list of the following structure and these specific estimates are produced using the simulated data set:

```
# should take 15-30 seconds
> starting.values <- c(-2, 2)
> frog<-nlm(intlik1,starting.values,y=y,delta=.1,X=traplocs,ssbuffer=2,hessian=TRUE)
> frog

$minimum
[1] 297.1896

$estimate
[1] -2.504824  2.373343

$gradient
[1] -2.069654e-05  1.968754e-05

$hessian
      [,1]      [,2]
[1,] 48.67898 -19.25750
[2,] -19.25750 13.34114

$code
[1] 1

$iterations
[1] 11
```

Details about this output can be found on the help page for `nlm`. We note briefly that `frog$minimum` is the negative log-likelihood value at the MLEs, which are stored in the `frog$estimate` component of the list. The hessian is the observed Fisher information matrix, which can be inverted to obtain the variance-covariance matrix using the commands:

```
> solve(frog$hessian)
```

It is worth drawing attention to the fact that the estimates are different than the Bayesian estimates reported in the previous chapter (section XYZ)!!! How can that be?! There are several reasons for this. First Bayesian inference is based on

the posterior distribution and it is not generally the case that the MLE should correspond to any particular value of the posterior distribution. If the prior distributions in a Bayesian analysis are uniform, then the mode of the posterior is the MLE, but note that Bayesians almost always report posterior means and so there will typically be a discrepancy there. Secondly, we have implemented an approximation to the integral here and there might be a slight bit of error induced by that. We will evaluate that shortly. Third, the Bayesian analysis by MCMC is subject to some amount of Monte Carlo error which the analyst should always be aware of in practical situations. All of these different explanations are likely responsible for some of the discrepancy. Accounting for these, as a practical matter, we see general consistency between the two estimates.

To compute the integrated likelihood we used a discrete representation of the state-space so that the integral could be approximated as a summation over possible values of \mathbf{s} with each value being weighted by its probability of occurring, which is $1/nG$ under the assumption that \mathbf{s} is uniform on the state-space \mathcal{S} . In chapter 4 we used a discrete state-space in developing a Bayesian analysis of the model in order to be able to modify the state-space in a flexible manner. Bayesian analysis requires simulation of the point process conditional on the observations, and this can be a difficult task when the state-space is continuous but has irregular geometry. Conversely, if the state-space is a regular polygon then Bayesian analysis by MCMC is possibly more efficient with a continuous state-space. We emphasize that the state-space is a part of the model. In some cases there wont be a natural choice of state space beyond “some large rectangle containing the trap grid” and, in such cases, for regular detection functions the estimate of density is invariant to the size of the state-space (i.e., the buffer) as long as it is sufficiently large. However if there are good reasons to restrict the state-space, it will tend to have an influence on the likelihood and hence AIC and so forth. As an illustration, lets do that by changing the state space here.....Use my polygon clipping stuff

In summary, we note that, for the basic SCR model, integrated likelihood is a really easy calculation when N is known. Even for N unknown it is not too difficult, and we will do that shortly. However, if you can solve the known- N problem then you should be able to do a real analysis, for example by considering different values of N and computing the results for each value and then making a plot of the log-likelihood or AIC and choosing the value of N that produces the best log likelihood or AIC. As a homework problem we suggest that the reader take the code given above and try to estimate N without modifying the code by just repeatedly calling that code for different values of N and trying to deduce the best value. Nevertheless, we will formalize the unknown- N problem shortly. We note that the software package **DENSITY** (Efford et al., 2004) implements certain types of SCR models using integrated likelihood methods. **DENSITY** has been made into an **R** package called **secr** (Efford, 2011) and we provide an analysis of some data using **secr** shortly along with a discussion of its capabilities.

9.2 MLE WHEN N IS UNKNOWN

Here we build on the previous introduction to integrated likelihood but we consider now the case in which N is unknown. We will see that adapting the analysis based on the N -known model is really straightforward for the more general problem. The main distinction is that we don't observe the all-zero encounter history so we have to make sure we compute the probability for that encounter history which we do by tacking a row of zeros onto the encounter history matrix. In addition, we include the number of such all-zero encounter histories as an unknown parameter of the model. Call that unknown quantity n_0 . In addition, we have to be sure to include a combinatorial term to account for the fact that of the n observed individuals there are $\binom{N}{n}$ ways to realize a sample of size n . The combinatorial term involves the unknown n_0 and thus it must be included in the likelihood.

DETAILS NEEDED HERE

To summarize, when N is unknown, the n observed encounter histories have a multinomial distribution with probabilities $\pi(i)$ and sample size N^2 . The last cell the “zero cell” is computed by carrying out the integral in expression XYZ above for the all-zero encounter history and we have to account for the fact that there are $n_0 = N - n$ such encounter histories.

To analyze a specific case, we'll read in our fake data set (simulated using the parameters given above). To set some things up in our workspace we do this:

```
data<-simSCR0.fn(discard0=TRUE,sd=2013)
y<-data$Y
nind<-nrow(y)
X<-data$traplocs
J<-nrow(X)
K<-data$K
Xl<-data$xlim[1]
Yl<-data$ylim[1]
Xu<-data$xlim[2]
Yu<-data$ylim[2]
```

Recall that these data were generated with $N = 100$, on an 8×8 unit state-space representing the trap locations (\mathbf{X}) buffered by 2 units. As before, the likelihood is defined in the **R** workspace as an **R** function which takes an argument being the unknown parameters of the model and additional arguments as prescribed. In particular, as before, we provide the encounter history matrix \mathbf{y} , the trap locations traplocs , the spacing of the integration grid (δ) and the state-space buffer. Here is the new likelihood function:

```
intlik2<-function(parm,y=y,delta=.3,X=traplocs,ssbuffer=2){
```

² Maybe you could show an alternative simulation script to generate data using the `rmultinom` function. This would make it a little more clear for people


```

4334
4335 Xl<-min(X[,1]) -ssbuffer
4336 Xu<-max(X[,1])+ ssbuffer
4337 Yu<-max(X[,2])+ ssbuffer
4338 Yl<-min(X[,2]) - ssbuffer
4339
4340 #delta<- (Xu-Xl)/npix
4341 xg<-seq(Xl+delta/2,Xu-delta/2,delta)
4342 yg<-seq(Yl+delta/2,Yu-delta/2,delta)
4343 npix.x<-length(xg)
4344 npix.y<-length(yg)
4345 area<- (Xu-Xl)*(Yu-Yl)/((npix.x)*(npix.y))
4346 G<-cbind(rep(xg,npix.y),sort(rep(yg,npix.x)))
4347 nG<-nrow(G)
4348 D<- e2dist(X,G)
4349
4350 alpha0<-parm[1]
4351 alpha1<-parm[2]
4352 n0<-exp(parm[3])
4353 probcap<- plogis(alpha0)*exp(-alpha1*D*D)
4354 Pm<-matrix(NA,nrow=nrow(probcap),ncol=ncol(probcap))
4355 ymat<-rbind(y,rep(0,ncol(y)))
4356
4357 lik.marg<-rep(NA,nrow(ymat))
4358 for(i in 1:nrow(ymat)){
4359   Pm[1:length(Pm)]<- (dbinom(rep(ymat[i,],nG),K,probcap[1:length(Pm)],log=TRUE))
4360   lik.cond<- exp(colSums(Pm))
4361   lik.marg[i]<- sum( lik.cond*(1/nG) )
4362 }
4363 nv<-c(rep(1,length(lik.marg)-1),n0)
4364 part1<- lgamma(nrow(y)+n0+1) - lgamma(n0+1)
4365 part2<- sum(nv*log(lik.marg))
4366 -1*(part1+ part2)
4367 }

```

4368 To execute this function for the data that we created with `simSCRO.fn`, we
 4369 execute the following command (saving the result in our friend `frog`). This re-
 4370 sults in the usual output, including the parameter estimates, the gradient, and the
 4371 numerical Hessian which is useful for obtaining asymptotic standard errors (see
 4372 below):

```

4373 > frog<-nlm(intlik2,c(-2.5,2,log(4)),hessian=TRUE,y=y,X=X,delta=.2,ssbuffer=2)
4374 There were 50 or more warnings (use warnings() to see the first 50)
4375 >

```

```

4376 >
4377 > frog
4378 $minimum
4379 [1] 113.5004
4380
4381 $estimate
4382 [1] -2.538334  2.466515  4.232810
4383
4384 [. Additional output deleted .]

```

While this produces some **R** warnings, these happen to be harmless in this case, and we will see from the **nlm** output that the algorithm performed satisfactory in minimizing the objective function. The estimate of population size for the state-space (using the default state-space buffer) is

```

4389 > nrow(y)+exp(4.2328)
4390 [1] 110.9099

```

Which differs from the data-generating value ($N = 100$) as we might expect. We usually will present an estimate of uncertainty associated with this MLE which we can obtain by inverting the Hessian. Note that $Var(\hat{N}) = n + Var(\hat{n}_0)$. Since we have parameterized the model in terms of $\log(n_0)$ we use a delta approximation to obtain the variance on the scale of n_0 as follows:

```

4396 > (exp(4.2328)^2)*solve(frog$hessian)[3,3]
4397 [1] 260.2033
4398 > sqrt(260)
4399 [1] 16.12452

```

9.2.1 Exercises

1. Run the analysis with different state-space buffers and comment on the result.
2. Conduct a brief simulation study using this code by simulating 100 data sets and obtain the MLEs for each data set. Do things seem to be working as you expect?
3. Further extensions: It should be straightforward to generalize the integrated likelihood function to accommodate many different situations. For examples, if we want to include more covariates in the model we can just add stuff to the object **probcap**, and add the relevant parameters to the argument that gets passed to the main function. For the simulated data, make up a covariate by generating a Bernoulli covariate (“trap type” perhaps baited or not baited) randomly and try to modify the likelihood to accommodate that.

4411 4. We would probably be interested in devising the integrated likelihood for the full
4412 3-d encounter history array so that we could include temporally varying covariates.
4413 This is not difficult but naturally will slow down the execution substantially. The
4414 interested reader should try to expand the capabilities of this basic **R** function.

4415 9.2.2 Integrated Likelihood using the model under data augmentation

4416 Note that this likelihood analysis is based on the standard likelihood in which N
4417 (or n_0) is an explicit parameter. This is usually called the “joint likelihood” or
4418 “unconditional likelihood”. We could also express the joint likelihood using data
4419 augmentation, replacing the parameter N with ψ (e.g., Royle and Dorazio, 2008,
4420 sec. xyz). We don’t go into detail here, but we do note that the likelihood under
4421 data augmentation is a zero-inflated binomial mixture precisely as an occupancy
4422 type model (Royle, 2006). The interested reader could adapt the material from
4423 Royle and Dorazio (2008) with the **R** code given above for the likelihood and
4424 implement the likelihood analysis based on the model under data augmentation.
4425 While we can carryout likelihood analysis of models under data augmentation, we
4426 primarily advocate data augmentation for Bayesian analysis.

4427 9.2.3 Extensions

4428 There are other types of covariates of interest: behavioral response, sex-specificity
4429 of parameters and all of these things. Some of these can be added directly to
4430 the likelihood if the covariate is fixed and known for all individuals captured or
4431 not. This excludes most covariates but it does include behavioral response. Sex-
4432 specificity is more difficult since sex is not known for uncaptured individuals. Trap-
4433 specific covariates such as trap type or status, or time-specific covariates such as
4434 date, are relatively easy to deal with (we leave these as exercises). We apply these
4435 various models in Chapter XXXX. To analyze such models, we do Bayesian analysis
4436 of the joint likelihood facilitated by the use of data augmentation. For covariates
4437 that are not fixed and known for all individuals, it is hard to do MLE for these based
4438 on the joint likelihood as we have developed above. Instead what people normally
4439 do is use what is colloquially referred to as the “Huggins-Alho” type model which
4440 is one of the approaches taken in the software package **secr** (Efford, 2011, see sec.
4441 9.5).

9.3 CLASSICAL MODEL SELECTION AND ASSESSMENT

4442 In most analyses, one is interested in choosing from among various potential mod-
4443 els. A good thing about classical analysis based on likelihood is we can do rote
4444 application of AIC without thinking about anything. With distance as a covariate
4445 (e.g., distance sampling) this is usually applied to some arbitrary selection of dis-
4446 tance functions. We dont recommend this. Given there is hardly ever (if at all) a

rational science-based reason for choosing some particular distance function we believe that this standard approach will invariably lead to over-fitting. The fact that AIC is easy to compute does not mean that it should be abused in such fashions. Further discussion is made in chapters XYZ.

Goodness-of-fit In many analyses based on likelihood methods it is possible to cook-up fit statistics for which asymptotic distributions are unknown. In general, however, applied statisticians tend to adopt bootstrapping based on heuristically appealing fit statistics. An omnibus global GoF statistic is not so obvious but we can apply bootstrapping principles to SCR models directly which we discuss in chapter XYZ. Bayesian goodness-of-fit is almost always addressed with Bayesian p-values or some other posterior predictive check (REF XXX). Thus the approach whether Bayesian or classical is the same. We identify a fit statistic, we do a bootstrap (classical) or a Bayesian p-value. Royle et al. (2011) decomposed the fit problem into separate evaluations of the CSR hypothesis and the encounter process model. We discuss all of this in Chapter XYZ.

9.4 LIKELIHOOD ANALYSIS OF THE WOLVERINE CAMERA TRAPPING DATA

Here we compute the MLEs for the wolverine data using an expanded version of the function we developed in the previous section. To accommodate that each trap might be operational a variable number of nights, we provided an additional argument to the likelihood function (allowing for a vector K), which requires also a modification to the construction of the likelihood. In addition, we had to accommodate that the state-space is a general rectangle, and we included a line in the code to compute the state-space area which we apply below for computing density. The more general function (`intlik3`) is given in the **R** package. It has a general purpose wrapper named `scr` which has other capabilities too.

The data were read into our R session and manipulated using the following commands. Note that we use the utility **R** function `SCR23darray.fn` which we defined in chapt. 4.

```

> wcaps<-source("wcaps.R")$value
> wtraps<-source("wtraps.R")$value
> K.wolv<-apply(wtraps[,4:ncol(wtraps)],1,sum)
>
> xx<-SCR23darray.fn(wcaps,ntraps=37,nperiods=165)
> y.wolv<- apply(xx,c(1,3),sum)
> traplocs.wolv<-wtraps[,2:3]
> traplocs.wolv<-traplocs.wolv/10000
>
> frog<-nlm(intlik3,c(-1.5,1.2,log(4)),hessian=TRUE,y=y.wolv,K=K.wolv,X=traplocs.wolv,delta=.1,s
There were 23 warnings (use warnings() to see them)

```

```

4485 > frog
4486
4487 $minimum
4488 [1] 220.4355
4489
4490 $estimate
4491 [1] -2.817570  1.255112  3.599040
4492
4493 $gradient
4494 [1] -6.274309e-06  2.146722e-05 -1.045566e-05
4495
4496 $hessian
4497           [,1]      [,2]      [,3]
4498 [1,] 37.687931 -11.852236  4.688911
4499 [2,] -11.852236 30.846144 -9.199113
4500 [3,]  4.688911 -9.199113 13.050428
4501
4502 $code
4503 [1] 1
4504
4505 $iterations
4506 [1] 12
4507
4508 > exp(3.599)*sqrt(solve(frog$hessian)[3,3])
4509 [1] 11.41059
4510 >
4511

```

4512 We obtained the MLEs for a state-space buffer of 2 (standardized units) and
4513 for integration grid with spacing $\delta = .3, .2, .1, .05$. The MLEs for these 4 cases
4514 including the relative runtime are given in Table 9.1.

Table 9.1. Run time and MLEs for different integration grid resolutions.

δ	runtime	Estimates		
		α_0	θ	$\log(n_0)$
0.30	8.4	-2.819786	1.258468	3.569731
0.20	22.6	-2.817610	1.254757	3.583690
0.10	99.0	-2.817570	1.255112	3.599040
0.05	403.0	-2.817559	1.255281	3.607158

4515 We see the results change only slightly as the fineness of the integration grid
4516 increases. Conversely, the runtime on the platform of the day for the 4 cases
4517 increases rapidly which, as we have suggested before, could probably be regarded

in relative terms, across platforms, for gaging the decrease in speed as the fineness of the integration grid increases. The effect of this is that we anticipate some numerical error in approximating the integral on a mesh of points, and that error increases as the coarseness of the mesh increases.

In section 6.9 back in chapt. 4 we used a discrete representation of the state-space in order to have control over its extent and shape, for example so that we could clip out “non-habitat”. Clearly that formulation of the model is relevant to the use of integrated likelihood in the sense that such a representation of the state-space underlies the computation of the integral. Thus, for example, we could easily compute the MLE of parameters under some model with a restricted state-space merely by creating the required state-space at whatever grid resolution is desired, and then feed that state-space into the likelihood evaluation above. The **R** function `scr` which comes with the **R** package for this book accommodates an arbitrary state-space fashioned in this manner, as well as state-spaces created by polygons or GIS shapefiles³.

Next we studied the effect of the state-space buffer on the MLEs, using a fixed $\delta = .2$ for all analyses. We used state-space buffers of 1 to 4 units stepped by .5. This produced the following results, given here are the state-space buffer, area of the state-space, the MLE of N for the prescribed state-space and the corresponding MLE of density:

	ssbuff	Ass	Nhat	Dhat
[1,]	1.0	66.98212	37.73338	0.5633352
[2,]	1.5	84.36242	46.21008	0.5477567
[3,]	2.0	103.74272	57.00617	0.5494956
[4,]	2.5	125.12302	69.03616	0.5517463
[5,]	3.0	148.50332	82.17550	0.5533580
[6,]	3.5	173.88362	96.44018	0.5546249
[7,]	4.0	201.26392	111.83524	0.5556646

The estimates of D stabilize rapidly and the incremental difference is within the numerical error associated with approximating the integral. The results suggest that wolverine density is around 0.56 individuals per 100 km^2 (recall that a state-space unit is $10 \times 10 km$). This is about 5.6 individuals per thousand km^2 which compares with XYZ-lookup-XYZ reported in Royle et al. (2011c) based on a clipped state-space as described in section XYZ (XYZ chapter 4 XYZ).

9.4.1 Exercises

1. Compute the 95% confidence interval for wolverine density, somehow.
2. Compute the AIC of this model and modify `intlik3` to consider alternative link

³to be completed!

4555 functions (at least one additional) and compare the AIC of the different models
 4556 and the estimates. Comment.

9.5 PROGRAM DENSITY AND THE R PACKAGE SECR

4557 **DENSITY** is a software program developed by Efford (2004) for fitting spatial
 4558 capture-recapture models based mostly on classical maximum likelihood estimation
 4559 and related inference methods. Efford (2011) has also released an **R** package named
 4560 **secr**, that contains many of the functions within **DENSITY** but also incorporates
 4561 new models and features. Here, we will focus on **secr** as it will continue to be
 4562 developed, contains more functionality and is based in **R**. To install and run models
 4563 in **secr**, you must download the package and load it in **R**.

```
4564 > install.packages(secr)
4565 > library(secr)
```

4566 **secr** allows the user to simulate data and fit a suite of models with various detection
 4567 functions and covariate responses. **secr** uses the standard **R** model specification
 4568 framework using tildes. E.g., the model command is **secr.fit** and is generally
 4569 written as

```
4570 > secr.fit(capturedata, model = list(D~1, g0~1, sigma~1), buffer = 20000)
```

4571 where we have **g0~1** indicating the intercept model. To include covariates, this
 4572 would be written as **g0~b** where *b* is a behavioral response covariate. Possible
 4573 predictors for detection probability include both pre-defined variables (e.g., **t** and
 4574 **b** corresponding to “time” and “behavior”), and user-defined covariates of several
 4575 kinds. The discussion of covariates is developed in chapter XX(8)⁴

4576 Before we can fit the models, the data must first be entered into **secr**. Two
 4577 input files are required: trap layout (location and identification information for
 4578 each trap) and capture data (e.g., sampling session, animal identification, trap
 4579 day, and trap location). **SECR** requires that you specify the trap type, the two
 4580 most common for camera trapping/hair snares are proximity detectors and count
 4581 detectors. The ‘proximity’ detector type allows, at most, one detection of each
 4582 individual at a particular detector on any occasion. The count detector designation
 4583 allows repeat encounters of each individual at a particular detector on any occasion.
 4584 There are other detector types that one can select such as: ‘polygon’ detector type
 4585 which allows for a trap to be a sampled polygon, e.g., scat surveys, and ‘signal’
 4586 detector which allows for traps that have a strength indicator, e.g., acoustic arrays.
 4587 The detector types single and multi can be confusing as multi seems like it would
 4588 appropriate for something like a camera trap, but instead these two designations
 4589 refer to traps that retain individuals, thus precluding the ability for animals to be
 4590 captured in other traps during the sampling occasion. The single type indicates

⁴Beth: does secr fit a local trap-specific response or just a global behavioral response?

4591 trap that can only catch one animal at a time, while multi indicates traps that may
 4592 catch more than one animal at a time. For a full review of the detector types, one
 4593 should look at the help manual, which can be accessed in R after installing the
 4594 SECR package by using the command:

```
4595 > RShowDoc("secr-manual", package = "secr")
```

4596 As with all of the secr models, **secr** fits a detection function relating the proba-
 4597 bility of detection to the distance of a detector from an individual activity center.
 4598 **secr** allows the user to specify one of a variety of detection functions including the
 4599 commonly used half-normal, hazard rate, and exponential. There are 12 different
 4600 functions, but some are only available for simulating data, and one should take
 4601 caution when using different detection functions as the interpretation of the pa-
 4602 rameters, such as sigma, may not be consistent across formulations. The different
 4603 detection functions are defined in the secr manual and can be found by calling the
 4604 help function for the detection function:

```
4605 > ?detectfn
```

4606 It is useful to note that **secr** requires the buffer distance to be defined in meters
 4607 and density will be returned as number of animals per hectare. Thus to make
 4608 comparisons between **secr** and other models, we will often have to convert the
 4609 density to the same units. Also, note that sigma is returned in units of meters.

5

4611 9.5.1 Analysis using the secr package

4612 To demonstrate the use of the secr package, we will show how to do the same
 4613 analysis on the wolverine study as shown in section 4.6. To use the secr package,
 4614 the data need to be formatted in a similar but slightly different manner than we
 4615 use in WinBUGS. After installing the secr package, we first have to read in the trap
 4616 locations and other related information, such as if the trap is operational during a
 4617 sampling occasion. The secr package reads in the trap data through a command
 4618 called read.traps, which requires the detector type as input. The detector type
 4619 is important because it will determine the likelihood that secr will use to fit the
 4620 model. Here, we have selected proximity since individuals are captured at most
 4621 once in each trap during each sampling occasion.

```
4622 > traps= read.csv(wtraps.csv)
4623 > colnames(traps)[1:3]<- c("trapID","x", "y") #name the first 3 columns
4624 # to match the secr nomenclature
```

⁵One question: SECR only ever reports sigma. What exactly is sigma? It is a scale parameter of a detection function and all detection functions have a scale parameter. But in what sense is this sigma parameter related to home range diameter? Efford doesn't explain this, does he? In some sections in chapter 4 or possibly 6 we get into this issue.


```

4625
4626 > trapfile <- read.traps(data = traps, detector = "proximity")

4627     After reading in the data, we now need to create the encounter matrix or array.
4628     The secr package does this through the use of the make.caphist command, where
4629     we provide the capture histories in raw data format (each line contains the session,
4630     identification number, occasion, and trap id for only 1 individual). This is the
4631     format that was shown in the data input file wcaps, and we only need a line or
4632     two to organize the data into the order that the make.caphist command wants. In
4633     creating the capture history, we provide also the trapfile with the trap information,
4634     and the format (e.g., here fmt= trapID) so that secr knows how to match the
4635     encounters to the trap, and finally, we provide the number of occasions.

4636 > wolv.dat <- wcaps[,c(2, 3, 1)]
4637     #NEED TO UPDATE THIS WHEN I GET THE FILES,
4638     ### I JUST GUESSED AT THE CODE, BUT WOULD LIKE TO TRY IT.
4639 > wolv.dat <- cbind(rep(1, dim(wolv.dat)[1]), wolv.dat)
4640 > colnames(wolv.dat) <- c("Session", "ID", "Occasion", "trapID")
4641
4642 > wolvcapt=make.caphist(wolv.dat, trapfile, fmt = "trapID", noccasions = 165)

4643     Calling the secr.fit command, will run the model. We are using the basic model
4644     (SCR0), so we do not need to make any specifications in the command line except
4645     for the providing the buffer size (in m). To specify different models, you can change
4646     the default D~1, g0~1, sigma~1, which the interested reader can do with very little
4647     difficulty.

4648 > wolv.secr=secr.fit(wolvcapt, model = list(D~1, g0~1, sigma~1), buffer = 20000)
4649
4650 > wolv.secr
4651
4652 secr.fit( caphist = wolvcapt, buffer = 20000, binomN = 1 )
4653 secr 2.0.0, 18:26:39 05 Jul 2011
4654
4655 Detector type      proximity
4656 Detector number    37
4657 Average spacing    4415.693 m
4658 x-range            593498 652294 m
4659 y-range            6296796 6361803 m
4660 N animals          : 21
4661 N detections       : 115
4662 N occasions        : 165
4663 Mask area          : 1037069 ha
4664

```

```

4665 Model          : D~1 g0~1 sigma~1
4666 Fixed (real)    : none
4667 Detection fn     : halfnormal
4668 Distribution      : poisson
4669 N parameters     : 3
4670 Log likelihood    : -746.754
4671 AIC              : 1499.508
4672 AICc             : 1500.920
4673
4674 Beta parameters (coefficients)
4675      beta      SE.beta      lcl      ucl
4676 D      -9.749576 0.23027860 -10.200913 -9.298238
4677 g0     -4.275736 0.15846104  -4.586313 -3.965158
4678 sigma  8.699202 0.07868944   8.544973  8.853430
4679
4680 Variance-covariance matrix of beta parameters
4681      D      g0      sigma
4682 D      0.053028233 0.000546922 -0.005226926
4683 g0      0.000546922 0.025109900 -0.005885213
4684 sigma -0.005226926 -0.005885213 0.006192027
4685
4686 Fitted (real) parameters evaluated at base levels of covariates
4687      link      estimate SE.estimate      lcl      ucl
4688 D      log 5.831941e-05 1.360973e-05 3.713638e-05 9.158548e-05
4689 g0     logit 1.371121e-02 2.142902e-03 1.008756e-02 1.861207e-02
4690 sigma  log 5.998124e+03 4.727205e+02 5.140849e+03 6.998355e+03

```

4691 Under the fitted (real) parameters, we find D , the density, given in units of in-
4692 dividuals/hectare (1 hectare = 100 m²). To convert this into individuals/1000km²,
4693 we multiply by 100000, thus our density estimate is 5.83 individuals/1000 km².
4694 Sigma is given in units of meters, to convert to kilometers, we divide by 1000,
4695 which puts sigma at 5.99 km. Both of these estimates are very similar to those
4696 provided in section 4.6 for the buffer size equal to 20 km. As an exercise, run this
4697 analysis for 30 and 40 km buffers and compare those found in section 4.6 under
4698 **WinBUGS**. NOTE: The function `seccr.fit` will return a warning when the buffer
4699 size appears to be too small. This is useful particularly with the different units
4700 being used between programs and packages.

9.6 SUMMARY AND OUTLOOK

4701 In this chapter, we showed that classical analysis of SCR models based on likeli-
4702 hood methods is a relatively simple proposition. Analysis is based on the so-called
4703 integrated likelihood in which the individual activity centers (random effects) are

removed from the conditional-on- s likelihood by integration. We showed how to construct the integrated likelihood and fit some simple models in the R programming language. In addition, likelihood analysis for some broad classes of SCR models can be accomplished in the software package DENSITY or in the equivalent R library **secr** which we provided an illustration of here. In later chapters we provide more detailed analyses of SCR data using the **secr** package.

To compute the integrated likelihood we have to precisely describe the state-space of the underlying point process. In practice, this leads to a buffer around the trap array. We note that this is not really a buffer strip in the sense of Wilson et al. (XYZ) which is a feature of the analysis but it is somewhat more general here. In particular, it establishes the support of the integrand which we generally require to compute any integral. It might be that the integrand itself is finite even if the support is infinity but that may or may not be the case depending on the choice of detection function. As a practical matter then, it will typically be the case that, while estimates of N increase with the size of the buffer, estimates of density stabilize. This is not a feature of the classical methods based on using model M0 or model Mh and buffering the trap array.

Why or why not use likelihood inference exclusively? For certain specific models, it is probably more computationally efficient to produce MLEs. However, **WinBUGS** is extremely flexible in terms of describing models, although it sometimes can be quite slow. We can devise models in **WinBUGS** easily that we cannot fit in **secr**. E.g., random individual effects of various types (see next chapter), we can handle missing covariates in complete generality and seamlessly, and impose arbitrary distributions on random variables. Moreover, models can easily be adapted to include auxiliary data types. For example, we might have camera trapping and genetic data and we can describe the models directly in WinBUGS and fit a joint model. For the MLE we have to write a custom new piece of code for each model or hope someone has done it for us. Later we consider open population models which are straightforward to develop in WinBUGS but, so far, there is no available platform for doing MLE although we imagine one could develop this. Another thing that is more conceptual here is non-CSR point processes (see chapter XYZ) and generating predictions of how many individuals have home range centers in any particular polygon. Basic benefits of Bayesian analysis have been discussed elsewhere (Chapter 2? BPA book? Link and Barker?) and we believe these are compelling. On the other hand, likelihood analysis makes it easy to do model-selection by AIC. Goodness-of-fit is probably no more difficult or easy under either paradigm (see next chapter?).

In summary, basic SCR models are easy to implement by either likelihood or Bayesian methods but we feel that the typical user will realize much more flexibility in model development using existing platforms for Bayesian analysis. While these tend to be slow (sometimes excruciatingly slow), this will probably not be an impediment in most problems, especially at some near point in the future. Since we spent a lot of time here talking about specific technical details on how to implement

4747 likelihood analysis of SCR models, we provided a corresponding treatment in the
4748 next chapter on how to devise MCMC algorithms for SCR models. This is a bit
4749 more tedious and requires more coding, but is not technically challenging (except
4750 perhaps to develop highly efficient algorithms which we dont excel at).

4751
4752
4753

10

MCMC DETAILS

11

MCMC DETAILS

11.1 INTRODUCTION

In this chapter we will dive a little deeper into Markov chain Monte Carlo (MCMC) sampling. We will construct custom MCMC samplers in R, starting with easy-to-code GLMs and GLMMs and moving on to simple SCR models. We will also demonstrate some tricks and simple extensions to the 'spatial null model'. Finally, we will illustrate some alternative ready-to-use software packages for MCMC sampling. We will NOT provide exhaustive background information on the theory and justification of MCMC sampling there are entire books dedicated to that subject and we refer you to Robert and Casella (2004) and Robert and Casella (2010). Rather we aim to provide you with enough background and technical know-how to start building your own MCMC samplers for SCR models in R.

11.1.1 Why build your own MCMC algorithm?

The standard program we have used so far to run MCMC analyses is WinBUGS (Gilks et al., 1994). The wonderful thing about WinBUGS is that it will automatically use the most appropriate and efficient form of MCMC sampling for the model specified by the user.

The fact that we have such a Swiss Army knife type of MCMC machine begs the question: Why would anyone want to build their own MCMC algorithm? For one, there are a limited number of distributions and functions implemented in WinBUGS. While OpenBUGS provides more options, some more complex models may be impossible to build within these programs. A very simple example from spatial capture-recapture that can give you a headache in WinBUGS is when your state-space is an irregular-shaped polygon, rather than an ideal rectangle that can be characterized by four pairs of coordinates. It is easy to restrict activity centers

to any arbitrary polygon in R using an ESRI shapefile (and we will show you an example in a little bit), but you cannot use a shape file in a BUGS model.

Sometimes implementing an MCMC algorithm in R may be faster than in WinBUGS - especially if you want to run simulation studies where you have hundreds or more simulated data sets, several years' worth of data or other large models, this can be a big advantage.

Finally, building your own MCMC algorithm is a great exercise to understand how MCMC sampling works. So while using the BUGS language requires you to understand the structure of your model, building an MCMC algorithm requires you to think about the relationship between your data, priors and posteriors, and how these can be efficiently analyzed and characterized. Not to mention that, if you are an R junkie, it can actually be fun. However, if you don't think you will ever sit down and write your own MCMC sampler, consider skipping this chapter - apart from coding it will not cover anything SCR-related that is not covered by other, more model-oriented chapters as well.

11.2 MCMC AND POSTERIOR DISTRIBUTIONS

As mentioned in Chapter 2, MCMC is a class of simulation methods for drawing (correlated) random numbers from a target distribution, which in Bayesian inference is the posterior distribution. As a reminder, the posterior distribution is a probability distribution for an unknown parameter, say θ , given a set of observed data and its prior probability distribution (the probability distribution we assign to a parameter before we observe data). The great benefit of computing the posterior distribution of θ is that it can be used to make probability statements about θ , such as the probability that θ is equal to some value, or the probability that θ falls within some range of values. As an example, suppose we conducted a Bayesian analysis to estimate detection probability of some species at a study site (p), and we obtained a posterior distribution of $\text{beta}(20,10)$ for the parameter p . The following R commands demonstrate how we make inferences based upon summaries of the posterior distribution. Fig 1 shows the posterior along with the summary statistics.

```
> (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.

The posterior distribution summarizes all we know about a parameter and thus, is the central object of interest in Bayesian analysis. Unfortunately, in many if not most practical applications, it is nearly impossible to directly compute the posterior.

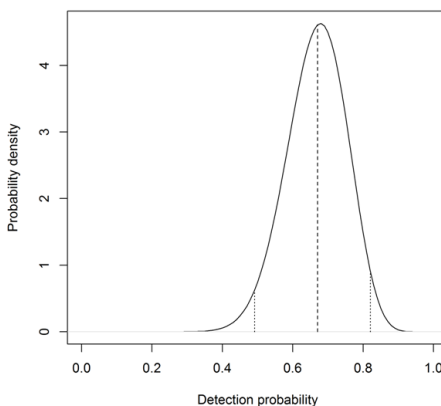


Figure 11.1. Probability density plot of a hypothetical posterior distribution of $\text{beta}(20,10)$; dashed lines indicate mean and upper and lower 95% interval

4818 Recall Bayes theorem:

$$p(\theta|y) = p(y|\theta) * p(\theta) / p(y), \quad (11.2.1)$$

4819 where θ is the parameter of interest, y is the observed data, $p(\theta|y)$ is the posterior,
 4820 $p(y|\theta)$ the likelihood of the data conditional on θ , $p(\theta)$ the prior probability of θ ,
 4821 and, finally, $p(y)$ is the marginal probability of the data, which can also be written
 4822 as

$$p(y) = \int p(y|\theta) * p(\theta) d\theta$$

4823 This marginal probability is a normalizing constant that ensures that the pos-
 4824 terior integrates to 1. You read in Chapter 2 that this integral is often hard or
 4825 impossible to evaluate, unless you are dealing with a really simple model. For ex-
 4826 ample, consider that you have a Normal model, with a set of n observations, y that
 4827 come from a Normal distribution:

$$y \sim \text{Normal}(\mu, \sigma),$$

4828 where σ is known and our objective is to obtain an estimate of μ using Bayesian
 4829 statistics. To fully specify the model in a Bayesian framework, we first have to
 4830 define a prior distribution for μ . Recall from Chapter 2 that for certain data
 4831 models, certain priors lead to conjugacy i.e. if you choose the right prior for your

parameter, your posterior distribution will be of a known parametric form. The conjugate prior for the mean of a normal model is also a Normal distribution:

$$\mu \sim \text{Normal}(\mu_0, \sigma_0^2)$$

If μ_0 and σ_0^2 are fixed, the posterior for μ has the following form (for the algebraic proof, see XXX):

$$\mu|y \sim \text{Normal}(\mu_n, \sigma_n^2) \quad (11.2.2)$$

where

$$\mu_n = (\text{sig}^2 / \text{sig}^2 + n * \text{sig}0^2) * \text{mu}0 + (n * \text{sig}0^2 / \text{sig}^2 + n * \text{sig}0^2) * y - \text{bar}$$

And

$$\text{sign}^2 = \text{sig}^2 * \text{sig}0^2 / (\text{sig}^2 + n * \text{sig}0^2)$$

We can directly obtain estimates of interest from this Normal posterior distribution, such as the mean μ -hat and its variance; we do not need to apply MCMC, since we can recognize the posterior as a parametric distribution, including the normalizing constant $p(y)$. But generally we will be interested in more complex models with several, say n , parameters. In this case, computing $p(y)$ from Eq. 11.2.1 requires n -dimensional integration, which is can be difficult or impossible. Thus, the posterior distribution is generally only known up to a constant of proportionality:

$$p(\theta|y) \propto p(y|\theta) * p(\theta)$$

The power of MCMC is that it allows us to approximate the posterior using simulation without evaluating the high dimensional integrals and to directly sample from the posterior, even when the posterior distribution is unknown! The price is that MCMC is computationally expensive. Although MCMC first appeared in the scientific literature in 1949 (Metropolis and Ulam, 1949), widespread use did not occur until the 1980s when computational power and speed increased (Gelfand and Smith, 1990). It is safe to say that the advent of practical MCMC methods is the primary reason why Bayesian inference has become so popular during the past three decades. In a nutshell, MCMC lets us generate sequential draws of θ (the parameter(s) of interest) from distributions approximating the unknown posterior over T iterations. The distribution of the draw at t depends on the value drawn at $t-1$; hence, the draws form a Markov chain.¹ As T goes to infinity, the Markov chain converges to the desired distribution. In our case the posterior distribution for θ — y . Thus, once the Markov chain has reached its stationary distribution, the generated samples can be used to characterize the posterior distribution, $p(\theta|y)$, and point estimates of θ , its standard error and confidence bounds, can be obtained directly from this approximation of the posterior. In practice, although we know that

¹In case you are not familiar with Markov chains, for t random samples $\theta(1), \dots, \theta(t)$ from a Markov chain the distribution of $\theta(t)$ depends only on the most recent value, $\theta(t-1)$.

a Markov chain will eventually converge, we can only generate a limited number of samples a process that depending on the model can be quite time consuming. Assessing whether our Markov chain has indeed converged is an important part of MCMC sampling and we will speak about some common diagnostics in Section XX.

11.3 TYPES OF MCMC SAMPLING

There are several MCMC algorithms, the most popular being Gibbs sampling and Metropolis-Hastings sampling. We will be dealing with these two classes in more detail and use them to construct the MCMC algorithms for SCR models. Also, we will briefly review alternative techniques that are applicable in some situations.

11.3.1 Gibbs sampling

Gibbs sampling was named after the physicist J.W. Gibbs by Geman and Geman (1984), who applied the algorithm to a Gibbs distribution². The roots of Gibbs sampling can be traced back to work of Metropolis et al. (1953), and it is actually closely related to Metropolis sampling (see Chapter 11.5 in Gelman et al. (2004), for the link between the two samplers). We will focus on the technical aspects of this algorithm, but if you find yourself hungry for more background, Casella and George (1992) provide a more in-depth introduction to the Gibbs sampler.

In Chapter 2 you already heard about the basic principles of Gibbs sampling³. But as a refresher, let's go back to our simple example from above to understand the motivation and functioning of Gibbs sampling. Recall that for a Normal model with known variance and a Normal prior for μ , the posterior distribution of $\mu|y$ is also Normal. Conversely, with a fixed (known) μ , but unknown variance, the conjugate prior for σ^2 is an Inverse-Gamma distribution with shape and scale parameters a and b :

$$\sigma^2 \sim \text{Inv-Gamma}(a, b),$$

With fixed a and b , the posterior $p(\text{sig}|\mu, y)$ is also an Inverse Gamma distribution, namely:

$$\text{sig}|\mu, y \sim \text{InvGamma}(an, bn), \quad (11.3.1)$$

where $an = n/2 + a$ and $bn = 1/2\sigma(y_i - \mu)^2 + b$. However, what if we know neither μ nor sig , which is probably the more common case? The joint posterior distribution of μ and sig now has the general structure

$$p(\mu, \text{sig}|y) = \frac{p(y|\mu) * p(\mu) * p(\text{sig})}{\int p(y|\mu) * p(\mu) * p(\text{sig}) d\mu d\text{sig}}$$

²a distribution from physics we are not going to worry about, since it has no immediate connection with Gibbs sampling other than giving its name

³maybe we should think out chapter 2 and concentrate that material here?

4891 Or

$$p(\mu, \sigma|y) \propto p(y|\mu) * p(\mu) * p(\sigma)$$

4892 This cannot easily be reduced to a distribution we recognize. However, we can
 4893 condition μ on σ (i.e., we treat σ as fixed) and remove all terms from the
 4894 joint posterior distribution that do not involve μ to construct the full conditional
 4895 distribution,

$$p(\mu|\sigma, y) \propto p(y|\mu) * p(\mu)$$

4896 The full conditional of μ again takes the form of the Normal distribution shown
 4897 in Eq. ??; similarly, $p(\sigma|\mu, y)$ takes the form of the Inverse Gamma distribution
 4898 shown in Eq. Eq. 11.3.1 both distribution we can easily sample from. And this is
 4899 precisely what we do when using Gibbs sampling we break down high-dimensional
 4900 problems into convenient one-dimensional problems by constructing the full con-
 4901 ditional distributions for each model parameter separately; and we sample from
 4902 these full conditionals, which, if we choose conjugate priors, are known parametric
 4903 distributions. Let's put the concept of Gibbs sampling into the MCMC framework
 4904 of generating successive samples, using our simple Normal model with unknown μ
 4905 and σ and conjugate priors as an example. These are the steps you need to build
 4906 a Gibbs sampler:

4907 **Step 0:** Begin with some initial values for θ , $\theta(0)$. In our example, we have to
 4908 specify initial values for μ and σ , for example by drawing a random number from
 4909 some uniform distribution, or by setting them close to what we think they might
 4910 be. (Note: This step is required in any MCMC sampling chains have to start from
 4911 somewhere. We will get back to these technical details a little later.)

4912 **Step 1:** Draw $\theta_1(1)$ from the conditional distribution $p(\theta_1(1) | \theta_2(0), \dots, \theta_d(0))$ Here,
 4913 θ_1 is μ , which we draw from the Normal distribution in Eq. ?? using $\sigma(0)$ as
 4914 value for σ .

4915 Step 2: Draw $\theta_2(1)$ from the conditional distribution $p(\theta_2(1) | \theta_1(1), \theta_3(0), \dots, \theta_d(0))$
 4916 Here, θ_2 is σ , which we draw from the Inverse Gamma distribution of Eq. 11.3.1,
 4917 using $\mu(1)$ as value for μ ...

4918 **Step d:** Draw $\theta_d(1)$ from the conditional distribution $p(\theta_d(1) | \theta_1(1), \dots, \theta_{d-1}(1))$

4919 In our example we have no additional parameters, so we only need step 0 through
 4920 to 2. Repeat Steps 1 to d for $K =$ a large number of samples. In terms of R coding,
 4921 this means we have to write Gibbs updaters for μ and σ and embed them into
 4922 a loop over K iterations. The final code in the form of an R function is shown in
 4923 Panel 1.

4924 Andy will build the panel environment here soon.

4925

4926 Panel 1: R-code for a Gibbs sampler for a Normal model with unknown μ

4927 and sig and conjugate (Normal and Inverse Gamma, respectively) priors
4928 for both parameters.

```
4929
4930 Normal.Gibbs<-function(y=y,mu0=mu0, sig0=sig0, a=a,b=b,niter=niter) {
4931
4932   ybar<-mean(y)
4933   n<-length(y)
4934   mu<-runif(1) #mean initial value
4935   sig<-runif(1) #sd initial value
4936   an<-n/2 + a
4937
4938   out<-matrix(nrow=niter, ncol=2)
4939   colnames(out)<-c('mu', 'sig')
4940
4941   for (i in 1:niter) {
4942
4943     #update mu
4944     mun<- (sig/(sig+n*sig0))*mu0 + (n*sig0/(sig+n* sig0))*ybar
4945     sign <- (sig*sig0)/ (sig+n*sig0)
4946     mu<-rnorm(1,mun, sqrt(sign))
4947
4948     #update sig
4949     bn<- 0.5 * (sum((y-mu)^2)) +b
4950     sig<-1/rgamma(1,shape=an, rate=bn)
4951     out[i,<-c(mu,sqrt(sig))
4952
4953   }
4954   return(out)
4955 }
```

4956 This is it! You can use the code `NormalGibbs.R` in the **R** package `scrbook` to
4957 simulate some data, $y \sim \text{Normal}(5, 0.5)$ and run your first Gibbs sampler. Your
4958 output will be a table with two columns, one per parameter, and K rows, one per
4959 iteration. For this 2-parameter example you can visualize the joint posterior by
4960 plotting samples of μ against samples of σ (Fig. 2 XXX):

```
4961 plot(out[,1], out[,2])
```

4962 The marginal distribution of each parameter is approximated by just examining the
4963 samples of this particular parameter you can visualize it by plotting a histogram
4964 of the samples (Fig. 3 a, b XXX):

```
4965 par(mfrow=c(1,2))
4966 hist(out[,1]); hist (out[,2])
```

Finally, recall an important characteristic of Markov chains, namely, that the chain has to have converged (reached its stationary distribution) for samples to come from the posterior distribution. In practice, that means you have to throw out some of the initial samples called the burn-in. We will talk about this in more when we talk about convergence diagnostics. For now, you can use the `plot(out[,1])` or `plot(out[,2])` command to make a time series plot of the samples of each parameter and visually assess how many of the initial samples you should discard. Figure 3 c and d shows plots for the estimates of mu and sigma from our simulated data set; you see that in this simple example the Markov chain apparently reaches its stationary distribution very quickly the chains look 'grassy' seemingly from the start. It is hard to discern a burn-in phase visually (but we will see examples further on where the burn-in is clearer) and you may just discard the first 500 draws to be sure you only use samples from the posterior distribution. The mean of the remaining samples are your estimates of mu and sig:

```
> summary(mod[501:10000,])
      mu          sig
Min.   : 4.936    Min.   : 0.4569
1st Qu.: 4.984    1st Qu.: 0.4889
Median : 4.994    Median : 0.4961
Mean   : 4.994    Mean   : 0.4964
3rd Qu.: 5.005    3rd Qu.: 0.5037
Max.   : 5.062    Max.   : 0.5356
```

11.3.2 Metropolis-Hastings sampling

Although it is applicable to a wide range of problems, the limitations of Gibbs sampling are immediately obvious what if we do not want to use conjugate priors (or what if we cannot recognize the full conditional distribution as a parametric distribution, or simply do not want to worry about these issues)? The most general solution is to use the Metropolis-Hastings (MH) algorithm, which also goes back to the work by Metropolis et al. (1953). You saw the basics of this algorithm in Chapter 2. In a nutshell, because we do not recognize the posterior $p(\theta|y)$ as a parametric distribution, the MH algorithm generates samples from a known proposal distribution, say $h(\theta)$, that depends on θ at $t-1$. The t^{th} sample is accepted or rejected based on its joint posterior probability density compared to the density of the sample at $t-1$. The original Metropolis algorithm requires $h(\theta)$ to be symmetric so that $h(\theta^t|\theta^{t-1}) = h(\theta^{t-1}|\theta^t)$; but a later development of the algorithm by Hastings (1970) lifted this condition. Using a symmetric proposal distribution makes life a little easier and we are going to limit our coverage of the Metropolis-Hastings sampler to this specific case. Specifically, we are going to use a Normal proposal distribution, which is also referred to as 'random walk Metropolis-Hastings

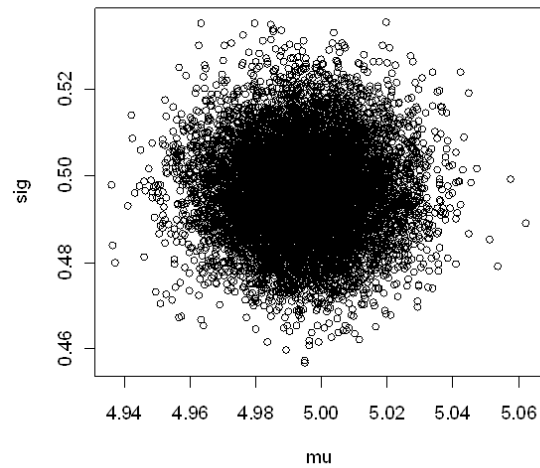


Figure 11.2. Joint posterior distribution of μ and σ from a Normal Model

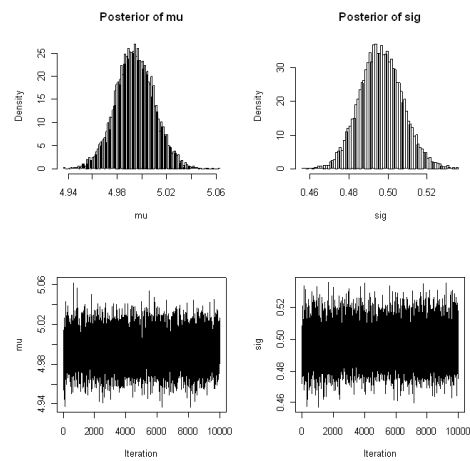


Figure 11.3. Plots of the posterior distributions of μ (a) and σ (b) from a Normal model and time series plots of μ (c) and σ (d).

sampling'. It is worth knowing that there are alternative formulations of the algorithm. For example, in the independent M-H, θ^t does not depend on θ^{t-1} , while the Langevin algorithm (Roberts and Rosenthal, 1998) aims at avoiding the random walk by favoring moves towards regions of higher posterior probability density. The interested reader should look up these algorithms in Robert and Casella (2004) or Robert and Casella (2010).

Building a MH sampler can be broken down into several steps. We are going to demonstrate these steps using a different but still simple and common model the logit-normal or logistic regression model. For simplicity, assume that

$$y \sim \text{Bern}(\exp(\theta)/(1 + \exp(\theta)))$$

and

$$\theta \sim \text{Normal}(\mu_0, \sigma)$$

The following steps are required to set up a random walk MH algorithm:

Step 0: Choose initial values, $\theta(0)$.

Step 1: Generate a proposed value of θ at t from $h(\theta^t - \theta^{t-1})$. We often use a Normal proposal distribution, so we draw θ_1 from $\text{Normal}(\theta^t, \text{sig}^2)$, where sig^2 is the variance of the Normal proposal distribution, a tuning parameter that we have to set.

Step 2: Calculate the ratio of posterior densities for the proposed and the original value for θ :

$$r = p(\theta^t|y)/p(\theta^{t-1}|y)$$

In our example,

$$r = \text{Bern}(y|\theta^t) * \text{Normal}(\theta^t|\mu_0, \sigma_0) / \text{Bernoulli}(y|\theta^{t-1}) * \text{Normal}(\theta^{t-1}|\mu_0, \text{sig}^2)$$

Step 3: Set

```
\begin{eqnarray*}
\theta(t) &=& \theta(t) \text{ with probability } \min(r,1) //
&=& \theta(t-1) \text{ otherwise }
\end{eqnarray*}
```

We can do that by drawing a random number u from a $\text{Unif}(0, 1)$ and accept θ^t if $u < r$. Repeat for $t = 1, 2, \dots$ a large number of samples. The **R** code for this MH sampler is provided in Panel 2 XXXX.

Panel 2: R code to run a Metropolis sampler on a simple Logit-Normal model.

```
Logreg.MH<-function(y=y, mu0=mu0, sig0=sig0, niter=niter) {
```

```

5037 out<-c()
5038
5039 theta<-runif(1, -3,3) #initial value
5040
5041 for (iter in 1:niter){
5042   theta.cand<-rnorm(1, theta, 0.2)
5043
5044   loglike<-sum(dbinom(y, 1, exp(theta)/(1+exp(theta)), log=TRUE))
5045   logprior <- dnorm(theta,mu0 ,sig0, log=TRUE)
5046   loglike.cand<-sum(dbinom(y, 1, exp(theta.cand)/(1+exp(theta.cand)), log=TRUE))
5047   logprior.cand <- dnorm(theta.cand, mu0, sig0, log=TRUE)
5048
5049   if (runif(1)<exp((loglike.cand+logprior.cand)-(loglike+logprior))){
5050     theta<-theta.cand
5051   }
5052   out[iter]<-theta
5053 }
5054
5055 return(out)
5056 }

```

5057 The reason we sum the logs of the likelihood and the prior, rather than multiply-
5058 ing the original values, is simply computational. The product of small probabilities
5059 can be numbers very close to 0, which computers do not handle well. Thus we add
5060 the logarithms, sum, and exponentiate to achieve the desired result. Similarly, in
5061 case you have forgotten some elementary math, $x/y = \exp(\log(x) - \log(y))$, with
5062 the latter being favored for computational reasons.

5063 Comparing MH sampling to Gibbs sampling, where all draws from the con-
5064 ditional distribution are used, in the MH algorithm we discard a portion of the
5065 candidate values, which inherently makes in less efficient than Gibbs sampling the
5066 price you pay for its increased generality. In Step 1 of the MH sampler we had to
5067 choose a variance for the Normal proposal distribution. Choice of the parameters
5068 that define our candidate distribution is also referred to as 'tuning', and it is im-
5069 portant since adequate tuning will make your algorithm more efficient, i.e. your
5070 Markov chain will converge faster. The variance should be chosen so that (a) each
5071 step of drawing a new proposal value for θ can cover a reasonable distance in the
5072 parameter space, as otherwise, the random walk moves too slowly; and (b) proposal
5073 values are not rejected too often, as otherwise the random walk will 'get stuck' at
5074 specific values for too long. As a rule of thumb, your candidate value should be ac-
5075 cepted in about 40% of all cases. Acceptance rates of 20-80% are probably ok, but
5076 anything below or above may well render your algorithm inefficient (this does not
5077 mean that it will give you wrong results only that you will need more iterations to
5078 converge to the posterior distribution). In practice, tuning will require some 'trial-
5079 and-error' and some common sense. Or, one can use an adaptive phase, where the
5080 tuning parameter is automatically adjusted until it reaches a user-defined accep-

5081 tance rate, at which point the adaptive phase ends and the actual Markov chain
 5082 begins. This is computationally a little more advanced. Link and Barker (2009)
 5083 discuss this in more detail. It is important the samples drawn during the adaptive
 5084 phase are discarded. You can easily check acceptance rates for the parameters you
 5085 monitor (that are part of your output) using the `rejectionRate()` function of the
 5086 package coda (we will talk more about this package a little later on). Do not let
 5087 the term 'rejection rate' confuse you; it is simply $1 - \text{acceptance rate}$. There may
 5088 be parameters for example, individual values of a random effect or latent variables
 5089 that you do not want to save, though, and in our next example we will show you a
 5090 way to monitor their acceptance rates with a few extra lines of code.

5091 11.3.3 Metropolis-within-Gibbs

5092 One weakness of the MH sampler is that formulating the joint posterior when
 5093 evaluating whether to accept or reject the candidate values for θ becomes increas-
 5094 ingly complex or inefficient as the number of parameters in a model increases. It
 5095 is probably going to sound like MCMC sampling is too good to be true but in
 5096 these cases you can simply combine MH sampling and Gibbs sampling. You can
 5097 use Gibbs sampling to break down your high-dimensional parameter space into
 5098 easy-to-handle one-dimensional conditional distributions and use MH sampling for
 5099 these conditional distributions. Better yet if you have some conjugacy in your
 5100 model, you can use the more efficient Gibbs sampling for these parameters and
 5101 one-dimensional MH for all the others. You have already seen the basics of how to
 5102 build both types of algorithms, so we can jump straight into an example here and
 5103 build a Metropolis-within-Gibbs algorithm.

11.4 GLMMS POISSON REGRESSION WITH A RANDOM EFFECT

5104 Let's assume a model that gets us closer to the problem we ultimately want to
 5105 deal with a GLMM. Here, we assume we have Poisson counts, y , from i plots
 5106 in j different study sites, and we believe that the counts are influenced by some
 5107 plot-specific covariate, x , but that there is also a random site effect. So our model
 5108 is:

$$y_{ij} \sim \text{Poisson}(l_{amij})$$

5109

$$l_{amij} = \exp(a_j + b * x_i)$$

5110 Let's use Normal priors on a and b ,

$$a_j \sim \text{Normal}(m_{ua}, s_{iga})$$

5111 and

$$b \sim \text{Normal}(m_{ub}, s_{igb})$$

5112 . ⁴ Since we want to estimate the random effect in this model, we do not specify
 5113 μ_a and σ_a , but instead, estimate them as well, so we have to specify hyperpriors
 5114 for these parameters:

$$\begin{aligned}\mu_a &\sim \text{Normal}(\mu_0, \text{sig}_0) \\ \sigma_a &\sim \text{InvGamma}(a_0, b_0)\end{aligned}$$

5115 With the model fully specified, we can compile the full conditionals, breaking
 5116 the multi-dimensional parameter space into one-dimensional components:

```
5117 \begin{eqnarray*}
5118 p(a1|a2,a3,aj,b,y) &\& \propto p(yi1|a1,b) * p(a1|mua, siga) \\\
5119 &\& \propto \text{Poisson}(yi1| \exp(a1 + b*x[j=1])) * \text{Normal}(a1|mua, siga) \\
5120 \end{eqnarray*}
5121 \begin{eqnarray*}
5122 p(a2|a1,a3,aj,b,y) &\& \propto p(yi2|a2,b) * p(a2|mua, siga) \\\
5123 &\& \propto \text{Poisson}(yi2|\exp(a2 + b*x[j=1])) * \text{Normal}(a2|mua, siga) \\
5124 \end{eqnarray*}
5125 \text{and so on for all elements of a.}
5126 \begin{eqnarray*}
5127 p(b|a,y) &\& \propto p(y|a,b) * p(b) \\\
5128 &\& \propto \text{Poisson}(y|\exp(a + b*x)) * \text{Normal}(b|mub, sigb) \\
5129 \end{eqnarray*}
```

5130 Finally, we need to update the hyperparameters for a:

$$\begin{aligned}5131 \quad p(mua|a) &\propto p(a|mua, siga) * p(mua) \\ p(siga|a) &\propto p(a|mua, siga) * p(siga)\end{aligned}$$

5132 Since we assumed a to come from a Normal distribution, the choice of priors for
 5133 mua Normal and siga Inverse Gamma leads to the same conjugacy we observed
 5134 in our initial Normal model, so that both hyperparameters can be updated using
 5135 Gibbs sampling.

5136 Now let's build the updating steps for these full conditionals. Again, for the
 5137 MH steps that update a and b we use Normal proposal distributions with standard
 5138 deviations sigha and sighb.

5139 First, we set the initial values a(0) and b(0). Then, starting with a1, we draw
 5140 a1(1) from Normal(a1(0), sigha), calculate the conditional posterior density of
 5141 a1(0) and a1(1) and compare their ratios,

$$r = \text{Poisson}(y(j=1)|\exp(a1(1)+b*x)) * \text{Normal}(a1(1)|mua, siga) / \text{Poisson}(y(j=1)|\exp(a1(0)+b*x)) * \text{Normal}(a1(0)|mua, siga)$$

5142 and accept a1(1) with probability min(r,1). We repeat this for all a's.

⁴Why is b a hyperparameter?

5143 For b, we draw $b(1)$ from $\text{Normal}(b(0), \text{sigbh})$, compare the posterior densities
5144 of $b(0)$ and $b(1)$,

$$r = \text{Poisson}(y|\exp(a+b(1)*x)) * \text{Normal}(b(1)|\text{mub}, \text{sigb}) / \text{Poisson}(y|\exp(a+b(0)*x)) * \text{Normal}(b(0)|\text{mub}, \text{sigb}),$$

5145 and accept $b(1)$ with probability $\min(r, 1)$.

5146 For mua and sig , we sample directly from the full conditional distributions (Eq
5147 XX and Eq XX):

$$\text{mua}(1) \sim \text{Normal}(\text{mun}, \text{sign})$$

5148 where $\text{mun} = (\text{sig}(0)/\text{sig}(0) + n_a * \text{sig0}) * \text{mu0} + (n_a * \text{sig0}/\text{sig}(0) + n_a * \text{sig0}) * \text{abar}(1)$ and $\text{sign} = \text{sig}(0) * \text{sig0}/(\text{sig}(0) + n * \text{sig0})$. Here, abar is the
5149 current mean of the vector a , which we updated before, and n_a is the length of
5150 a . For sig we use $\text{sig}(1) \sim \text{InvGamma}(an, bn)$, where $an = n_a/2 + a0$, and
5151 $bn = 1/2 \sum (\text{a}(1) - \text{mua}(1))^2 + b0$.
5152

5153 We repeat these steps over K iterations of the MCMC algorithm. In this example
5154 we may not want to save each value for a , but are only interested in their mean and
5155 standard deviation. Since these two parameters will change as soon as the value for
5156 one element in a changes, their acceptance rates will always be close to 1 and are
5157 not representative of how well your algorithm performs. To monitor the acceptance
5158 rates of parameters you do not want to save, you simply need to add a few lines
5159 of code into your updater to see how often the individual parameters are accepted.
5160 The full code for the MCMC algorithm of our Poisson GLMM in Panel 3 shows
5161 one way how to monitor acceptance of individual a 's.

5162 Panel 3: R code for the Metropolis-within-Gibbs sampler for
5163 a Poisson regression with random intercepts.

```
5164
5165 Pois.reg<-function(y=y,site=site,mu0=mu0,sig0=sig0,a0=a0,b0=b0,
5166                   mub=mub, sigb=sigb, niter=niter){
5167
5168   lev<-length(unique(site))      #number of sites
5169   a<-runif(lev,-5,5) #initial values a
5170   b<-runif(1,0,5) #initial value b
5171   mua<-mean(a)
5172   sig<-sd(a)
5173
5174   out<-matrix(nrow=niter, ncol=3)
5175   colnames(out)<-c('mua','sig','b')
5176
5177   for (iter in 1:niter) {
5178
5179     #update a
5180     aUps<-0 #initiate counter for acceptance rate of a
5181     for (j in 1:lev) { #loop over sites
5182       a.cand<-rnorm(1, a[j], 0.1) #update intercepts a one at a time
```

```

5183 loglike<- sum(dpois (y[site==j], exp(a[j] + b*x[site==j]), log=TRUE))
5184 logprior<- dnorm(a[j], mua,siga, log=TRUE)
5185 loglike.cand<- sum(dpois (y[site==j], exp(a.cand + b *x[site==j]), log=TRUE))
5186 logprior.cand<- dnorm(a.cand, mua,siga, log=TRUE)
5187 if (runif(1)< exp((loglike.cand+logprior.cand) (loglike+logprior))) {
5188   a[j]<-a.cand
5189   aUps<-aUps+1
5190 }
5191 }
5192
5193 if(iter %% 100 == 0) { #this lets you check the acceptance rate of a at every 100th iteration
5194   cat("   Acceptance rates\n")
5195   cat("     a =", aUps/lev, "\n")
5196 }
5197
5198 #update b
5199 b.cand<-rnorm(1, b, 0.1)
5200 avec<-rep(a, times=c(rep(10,10)))
5201 loglike<- sum(dpois (y, exp(avec + b*x), log=TRUE))
5202 logprior<- dnorm(b, mub,sigb, log=TRUE)
5203 loglike.cand<- sum(dpois (y, exp(avec + b.cand *x), log=TRUE))
5204 logprior.cand<- dunif(b.cand, mub,sigb, log=TRUE)
5205 if (runif(1)< exp((loglike.cand+logprior.cand) (loglike+logprior) )) {
5206   b<-b.cand
5207 }
5208
5209 #update mua using Gibbs sampling
5210 abar<-mean(a)
5211 mun<- (siga/(siga+lev*sig0))*mu0 + (lev*sig0/(siga+lev* sig0))*abar
5212 sign <- (siga*sig0)/ (siga+lev*sig0)
5213 mua<-rnorm(1,mun, sqrt(sign))
5214
5215 #update siga using Gibbs sampling
5216 a0n<-lev/2 + a0
5217 b0n<- 0.5 * (sum((a-mua)^2)) +b0
5218 siga<-1/rgamma(1,shape=a0n, rate=b0n)
5219
5220 out[iter,]<-c(mua, sqrt(siga), b)
5221
5222 }
5223
5224 return(out)
5225 }

```

11.4.1 Rejection sampling and slice sampling

While MH and Gibbs sampling are probably the most widely applied algorithms for posterior approximation, there are other options that work under certain circumstances and may be more efficient when applicable. WinBUGS applies these algorithms and we want you to be aware that there is more out there to approximate posterior distributions than Gibbs and MH. One alternative algorithm is rejection sampling. Rejection sampling is not an MCMC method, since each draw is independent of the others. The method can be used when the posterior $p(\theta|y)$ is not a known parametric distribution but can be expressed in closed form. Then, we can use a so-called envelope function, say, $g(\theta)$, that we can easily sample from, with the restriction that $p(\theta|y) < M * g(\theta)$. We then sample a candidate value for θ from $g(\theta)$, calculate $r = p(\theta|y) / M * g(\theta)$ and keep the sample with the probability r . M is a constant that has to be picked so that $r \in [0,1]$, for example by evaluating both $p(\theta|y)$ and $g(\theta)$ at n points and looking at their ratios. Rejection sampling only works well if $g(\theta)$ is similar to $p(\theta|y)$, and packages like WinBUGS use adaptive rejection sampling (Gilks and Wild, 1992), where a complex algorithm is used to fit an adequate and efficient $g(\theta)$ based on the first few draws. Though efficient in some situations, rejection sampling does not work well with high-dimensional problems, since it becomes increasingly hard to define a reasonable envelope function. For an example of rejection sampling in the context of SCR models, see Chapter 9. Another alternative is slice sampling (Neal, 2003). In slice sampling, we sample uniformly from the area under the plot of $p(\theta|y)$. Considering a single univariate θ . Let's define an auxiliary variable, $U \sim \text{Uniform}(0, p(\theta|y))$. Then, θ can be sampled from the vertical slice of $p(\theta|y)$ at U (Figure 4):

$$\theta|U \sim \text{Unif}(B),$$

where $B = \{\theta : U < p(\theta|y)\}$

⁵

Slice sampling can be applied in many situations; however, implementing an efficient slice sampling procedure can be complicated. We refer the interested reader to chapter 7 of Robert and Casella (2010) for a simple example. Both rejection sampling and slice sampling can be applied on one-dimensional conditional distributions within a Gibbs sampling setup.

11.5 MCMC FOR CLOSED CAPTURE-RECAPTURE MODEL MH

6

⁵there are supposed to be equations in the caption of figure 4 but it kept causing errors

⁶Andy could move material from chapter 3 to here.

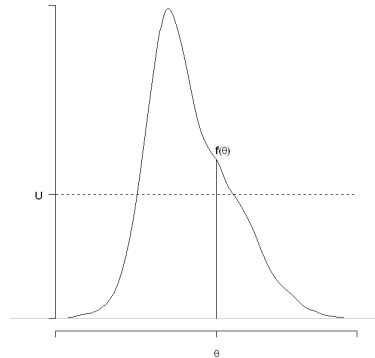


Figure 11.4. Slice sampling. For...

11.6 MCMC ALGORITHM FOR THE BASIC SPATIAL CAPTURE-RECAPTURE MODEL

By now you have seen how to build MCMC algorithms for some basic generalized linear models. Now, we'll walk you through the steps of building your own MCMC sampler for the basic SCR model (i.e. without any individual, site or time specific covariates) with both a Poisson and a binomial encounter process. As usual, we will have to go through two general steps before we write the MCMC algorithm:

- (1) Identify your model with all its components (including priors)
- (2) Recognize and express the full conditional distributions for all parameters

It is worthwhile to go through all of step 1 for an SCR model, but you have probably seen enough of step 2 in our previous examples to get the essence of how to express a full conditional distribution. Therefore, we will exemplify step 2 for some parameters and tie these examples directly to the respective R code.

Step 1 Identify your model

Recall the components of the basic SCR model with a Poisson encounter process from Chapter 3: We assume that individuals i , or rather, their activity centers s_i , are uniformly distributed across our state space S ,

$$s_i \sim U(S)$$

and that the number of times individual i encounters trap j , y_{ij} , is a random Poisson variable with mean λ_{mij} ,

$$y_{ij} \sim \text{Poisson}(\lambda_{mij})$$

The tie between individual location, movement and trap encounter rates is made by the assumption that λ_{mij} , is a decreasing function of the distance between s_i

5278 and j , D_{ij} , of the half-normal form

$$Lam_{ij} = lam0 * \exp(-D_{ij}^2/2 * sig^2),$$

5279 where $lam0$ is the baseline trap encounter rate at $D_{ij} = 0$ and sig controls the
5280 shape of the half-normal function.

5281 In order to estimate the number of s_i in S , N , we use data augmentation (sect.
5282 3.XYZ) and create $M-n$ all-0 encounter histories, where n is the number of individ-
5283 uals we observed and M is an arbitrary number that is larger than N . We estimate
5284 N by summing over the auxiliary data augmentation variables, z_i , which is 1 if the
5285 individual is part of the population and 0 if not, and assume that z_i is a random
5286 Bernoulli variable,

$$z_i \sim \text{Bern}(\psi)$$

5287 To link the two model components, we modify our trap encounter model to

$$Lam_{ij} = lam0 * \exp(-D_{ij}^2/2 * sig^2) * z_i.$$

5288 The model has the following structural parameters, for which we need to specify
5289 priors ψ the Uniform (0,1) is required as part of the data augmentation procedure
5290 and in general is a natural choice of an uninformative prior for a probability; note
5291 that this is equivalent to a Beta(1,1) prior, which will come in handy later. s_i since
5292 s_i is a pair of coordinates it is two-dimensional and we use a uniform prior limited
5293 by the extent of our state-space over both dimensions. σ we can conceive several
5294 priors for sigma but let's assume an improper prior one that is Uniform over (-Inf,
5295 Inf). We will see why this is convenient when we construct the full conditionals for
5296 sigma. λ_0 analogous, we will use a Uniform (-Inf, Inf) improper prior for sigma.
5297 The parameter that is the objective of our modeling, N , is a derived parameter that
5298 we can simply obtain by summing all z 's:

$$N = \text{sum}(z)$$

5299 **Step 2 - Construct the full conditionals** Having completed step 1, let's
5300 look at the full conditional distributions for some of these parameters. We find
5301 that with improper priors, full conditionals are proportional only to the likelihood
5302 of the observations; for example, take the movement parameter sigma:

$$Sig|s, lam0, z, ypropto[y|s, lam0, z, sig] * [sig]$$

5303 Since the improper prior implies that $[sig]$ propto 1, we can reduce this further to

$$Sig|s, lam0, z, ypropto[y|s, lam0, z, sig]$$

5304 The R code to update sigma is shown in Panel 4. ⁷

⁷ Somewhere in chapter 2 i added a comment about rejecting parameters outside of the parameter space as being an ok thing to do. Richard said he read something in Robert and Casellas book on that. Hopefully he can remember where and we can cite it back in Ch 2 and again here. It could be mentioned in a sentence or two up in the MCMC section.

5305 Panel 4: R code to update sigma within an MCMC algorithm for
 5306 an SCR model when using an improper prior

```
5307
5308
5309 sig.cand <- rnorm(1, sigma, 0.1) #draw candidate value
5310 if(sig.cand>0){ #automatically reject sig.cand that are <0
5311   lam.cand <- lam0*exp(-(D*D)/(2*sig.cand*sig.cand))
5312   ll<- sum(dpois(y, lam*z, log=TRUE))
5313   llcand <- sum(dpois(y, lam.cand*z, log=TRUE))
5314   if(runif(1) < exp( llcand - ll) ){
5315     ll<-llcand
5316     lam<-lam.cand
5317     sigma<-sig.cand
5318   }
5319 }
5320
```

5321 These steps are analogous for lam0 and si and we will use MH steps for all
 5322 of these parameters. Similar to the random intercepts in our Poisson GLMM, we
 5323 update each si individually. Note that to be fully correct, the full conditional for
 5324 si contains both the likelihood and prior component, since we did not specify an
 5325 improper, but a Uniform prior on si. However, with a Uniform distribution the
 5326 probability density of any value is 1/(upper limit - lower limit) = constant. Thus,
 5327 the prior components are identical for both the current and the candidate value
 5328 and can be ignored (formally, when you calculate the ratio of posterior densities, r,
 5329 the identical prior component appears both in the numerator and denominator, so
 5330 that they cancel each other out).

5331 We still have to update zi. The full conditional for zi is

$$z_i | y, \sigma, \lambda_0, \text{sprpto}[y | z, \sigma, \lambda_0, s] * [z_i]$$

5332 and since $z_i \sim \text{Bernoulli}(\psi_i)$, the term has to be taken into account when updating
 5333 zi. The R code for updating zi is shown in Panel 5.

5334 Panel 5: R code to update z

```
5335
5336 zUps <- 0 #set counter to monitor acceptance rate
5337 for(i in 1:M) {
5338   if(seen[i]) #no need to update seen individuals, since their z =1
5339     next
5340   zcand <- ifelse(z[i]==0, 1, 0)
5341   llz <- sum(dpois(y[i,], lam[i,]*z[i], log=TRUE))
5342   llcand <- sum(dpois(y[i,], lam[i,]*zcand, log=TRUE))
5343
5344   prior <- dbinom(z[i], 1, psi, log=TRUE)
5345   prior.cand <- dbinom(zcand, 1, psi, log=TRUE)

```

```

5346         if(runif(1) < exp( (llcand+prior.cand) - (llz+prior) )) {
5347             z[i] <- zcand
5348             zUps <- zUps+1
5349         }
5350     }

```

5351 ψ itself is a hyperparameter of the model, with an uninformative prior distribu-
 5352 tion of Unif(0,1) or Beta(1,1), so that

$$Psi|z \propto [z|psi] * Beta(1, 1)$$

5353 The Beta distribution is the conjugate prior to the Binomial and Bernoulli distribu-
 5354 tions (remember that $z \sim Bernoulli(psi)$). The general form of a full conditional
 5355 of a Beta-Binomial model with $y_i \sim Bernoulli(p)$ and $p \sim Beta(a, b)$ is

$$p(p|y) \propto Beta(a + sum(yi), b + n - sum(yi))$$

5356 In our case, this means we update psi as follows:

```

5357 si<-rbeta(1, 1+sum(z), 1 + M-sum(z))

```

5358 These are all the building blocks you need to write the MCMC algorithm for
 5359 the spatial null model with a Poisson encounter process. You can find the full R
 5360 code (SCR0pois.R) in the online supplementary material.

5361 11.6.1 SCR model with binomial encounter process

5362 The equivalent SCR model with a binomial encounter process is very similar. Here,
 5363 each individual i can only be detected once at any given trap j during a sampling
 5364 occasion k . Thus

$$y_{ij} \sim Binomial(p_{ij}, K)$$

5365 Where p_{ij} is some function of distance between s_i and trap location x_j . Here we
 5366 use:

$$p_{ij} = 1 - \exp(-lam_{ij})$$

5367 Recall from Chapter 2 that this is the complementary log-log (cloglog) link func-
 5368 tion, which constrains p_{ij} to fall between 0 and 1. For our MCMC algorithm that
 5369 means that, instead of using a Poisson likelihood, $Poisson(y|sigma, lam0, s, z)$, we
 5370 use a Binomial likelihood, $Binomial(y, K|sigma, lam0, s, z)$, in all the conditional
 5371 distributions. As an example, Panel 6 shows the updating step for $lam0$ under
 5372 a binomial encounter model. The full MCMC code for the binomial SCR can be
 5373 found in the online supplements.

5374 **Panel 6: MCMC updater for $lam0$ in a SCR model with Binomial encounter**
 5375 **process and cloglog link function on detection. Here, pmat =**

```

5376 1-exp(-lam).
5377
5378     lam0.cand <- rnorm(1, lam0, 0.1)
5379     if(lam0.cand > 0){ #automatically reject lam0.cand that are <0
5380         lam.cand <- lam0.cand*exp(-(D*D)/(2*sigma*sigma))
5381         p.cand <- 1-exp(-lam.cand)
5382         ll<- sum(dbinom(y, K, pmat *z, log=TRUE))
5383         llcand <- sum(dbinom(y, K, p.cand *z, log=TRUE))
5384         if(runif(1) < exp( llcand - ll) ){
5385             ll<-llcand
5386             pmat<-p.cand
5387             lam0<- lam0.cand
5388         }
5389     }

```

Another possibility is to model variation in the individual and site specific detection probability, p_{ij} , directly, without any transformation, such that

```

5392 pij<-p0 * exp(-Dij2/(2*sig^2))

```

and $p_0 = \{0,1\}$. This formulation is analogous to how detection probability is modeled in distance sampling under a half-normal detection function; however, in distance sampling p_0 - detection of an individual on the transect line - is assumed to be 1 (Buckland, 2001). Under this formulation the updater for lam0 (equivalent to p_0 in Eq XX) becomes:

```

5398     lam0.cand <- rnorm(1, lam0, 0.1)
5399     if(lam0.cand > 0 & lam0.cand < 1 ){ #automatically reject lam0.cand that are not
5400         lam.cand <- lam0.cand*exp(-(D*D)/(2*sigma*sigma))
5401         ll<- sum(dbinom(y, K, lam *z, log=TRUE)) #no transformation needed
5402         llcand <- sum(dbinom(y, K, lam.cand *z, log=TRUE))
5403         if(runif(1) < exp( llcand - ll) ){
5404             ll<-llcand
5405             lam<-lam.cand
5406             lam0<- lam0.cand
5407         }
5408     }

```

11.6.2 Looking at model output

Now that you have an MCMC algorithm to analyze spatial capture-recapture data with, let's run an actual analysis so we can look at the output. As an example, we will use the bear data ...⁸ You can use the same script provided back in

⁸Does this data set come up before Ch6? If not, introduce data here. Or, Andy, would you rather use simulated data?

Chapter XX to read in the data and build the augmented encounter history array; then source the MCMC code for the binomial encounter model algorithm with the cloglog link and run 5000 iterations. This should take approximately 25 minutes.

```
> source('SCR0binom.txt')
> mod0<-SCR.0(y=bigTrap, X=trapmat, M=M, xl=xl, xu=xu, yl=yl, yu=yu, K=8, niter=5000)
```

Before, we used simple R commands to look at model results. However, there is a specific R package to summarize MCMC simulation output and perform some convergence diagnostics package coda (Plummer et al., 2006). Download and install coda, then convert your model output to an mcmc object

```
> chain<-mcmc(mod0)
```

Markov chain time series plots

Start by looking at time series plots of your Markov chains using `plot(chain)`. This command produces a time series plot and marginal posterior density plots for each monitored parameter, similar to what we did before using the `hist()` and `plot()` commands (Fig. 5). Time series plots will tell you several things: First, the way the chains move through the parameter space gives you an idea of whether your MH steps are well tuned. If chains were constant over many iterations you would probably need to decrease the tuning parameter of the (Normal) proposal distribution. If a chain moves along some gradient to a stationary state very slowly, you may want to increase the tuning parameter so that the parameter space is explored more efficiently.

Second, you will be able to see if your chains converged and how many initial simulations you have to discard as burn-in. In the case of the chains shown in Figure 5, we would probably consider the first 750 - 1000 iterations as burn-in, as afterwards the chains seem to be fairly stationary.

A word of caution about chain convergence

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 approximation), we effectively have no means to assess whether it has converged to this desired distribution or not. As mentioned before, the only certainty is that a Markov chain will *eventually* converge to its stationary distribution, but no-one can tell us how long this will take. Also, you only now 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 proof 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. One option, and that advocated by what we will loosely call the WinBUGS community, is to run several Markov chains and to start them off at different initial values that are overdispersed relative to the posterior distribution. Such initial values help to explore different

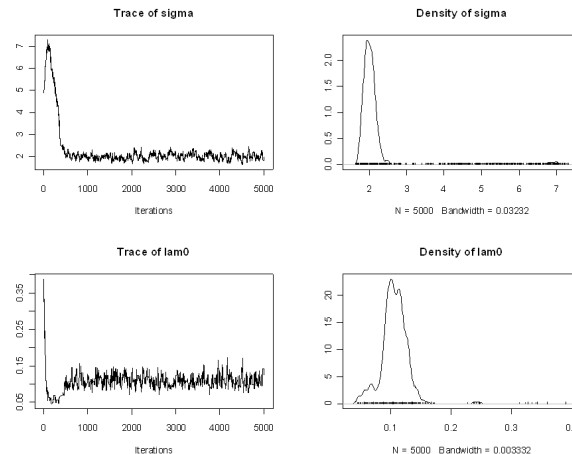


Figure 11.5. Time series and posterior density plots for sigma and lam0.

5453 areas of the parameter space simultaneously; if after a while all chains oscillate
 5454 around the same average value, chances are good that they indeed converged to
 5455 the posterior distribution. Gelman and Rubin came up with a diagnostic statistic
 5456 that essentially compares within-chain and between-chain variance to check for
 5457 convergence of multiple chains (Gelman et al., 2004). Of course, running several
 5458 parallel chains is computationally expensive. Extra computational demands are not
 5459 the only and by no means the major concern some people voice when it comes to
 5460 running several parallel MCMC chains to assess convergence. Again, consider the
 5461 fact that we do not know anything about the true form of the posterior distribution
 5462 we are trying to approximate. How do we, then, know how to pick overdispersed
 5463 initial values? We don't all we can do is pick overdispersed values relative to our
 5464 expectations of what the posterior should look like. To use a quote from the home
 5465 page of Charlie Geyer, a Bayesian statistician from the University of Minnesota,
 5466 "If you don't know any good starting points [...], then restarting the sampler at
 5467 many bad starting points is [...] part of the problem, not part of the solution."
 5468 (<http://users.stat.umn.edu/~charlie/mcmc/diag.html>). His suggestion is that your
 5469 only chance to discover a potential problem with your MCMC sampler is to run it
 5470 for a very long time. But again, there is no way of knowing how long is long enough.
 5471 It is up to you to decide, which school of thoughts appeals more to you one long
 5472 versus several parallel Markov chains. Irrespectively, part of developing an MCMC
 5473 sampler should be to make sure (within reasonable limits) that you are not missing
 5474 regions of high posterior density because of the way you specify your starting values.
 5475 Once you have explored the behavior of your chain under a reasonable range of

starting values, you may feel comfortable enough to run only one long chain. The fact that convergence cannot be proven does not mean that you should not look for potential problems in your MCMC sampler. Some problems are easily detected using simple plots, such as the time series plots we discussed above. If the overall trajectory of your chain at the end of your simulations is still upward or downward, your chain clearly has not converged and you need to run your model much longer. If you run several parallel chains and their stationary distributions look different, you may be looking at a multi-modal posterior or a problem with your sampler. With these words of caution, let's get back to looking at our model output.

11.6.3 Posterior density plots

The `plot()` command also produces posterior density plots and it is worthwhile to look at those carefully. For parameters with priors that have bounds (e.g. Uniform over some interval), you will be able to see if your choice of the prior is truncating the posterior distribution. In the context of SCR models, this will mostly involve our choice of M , the size of the augmented data set. If the posterior of N has a lot of mass concentrated close to M (or equivalently the posterior of ψ has a lot of mass concentrated close to 1), as in the example in Figure 6, we have to re-run the analysis with a larger M . A flat posterior plot shows you that the parameter essentially cannot be identified there may not be enough information in your data to estimate model parameters and you may have to consider a simpler model. Finally, posterior density plots will show you if the posterior distribution is symmetrical or skewed if the distribution has a heavy tail, using the mean as a point estimate of your parameter of interest may be biased and you may want to opt for the median or mode instead.

11.6.4 Serial autocorrelation and effective sample size

Even when we can be relatively confident that our chains have converged, the subsequent samples generated from a Markov chain are not iid samples from the posterior distribution, due to the correlation amongst samples introduced by the Markov process. As a consequence, the variance of the mean cannot simply be derived with the standard variance estimator, which takes into account the sample size (here, number of iterations). Rather, the sample size has to be adjusted to account for the autocorrelation in subsequent samples (see Chapter 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. If you use WinBUGS through the R2WinBUGS package, the `print()` command will automatically return the effective sample size for all monitored parameters. In the coda package there are several functions you

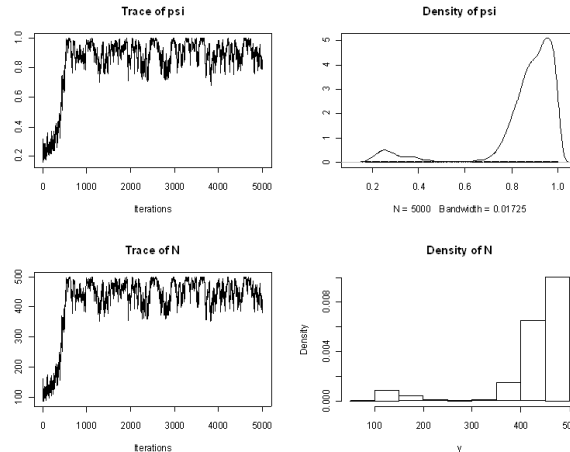


Figure 11.6. Time series and posterior density plots of ψ and N for the bear data set truncated by the upper limit of M (500).

5514 can use to do so. `effectiveSize()` will directly give you an estimate of the effective
5515 sample size for your parameters:

```
5516 > effectiveSize(chain)
5517     sigma     lam0     psi      N
5518 3.930303 78.259159 30.436348 32.047392
```

5519 Alternatively, you can use the `autocorr.diag()` function, which will show you the
5520 degree of autocorrelation for different lag values (which you can specify within the
5521 function call, we use the defaults below):

```
5522 > autocorr.diag(mcmc(mod))
5523     sigma     lam0     psi      N
5524 Lag 0  1.0000000 1.0000000 1.0000000 1.0000000
5525 Lag 1  0.9979948 0.9494134 0.9847503 0.9774201
5526 Lag 5  0.9915567 0.8038168 0.9111951 0.9113525
5527 Lag 10 0.9836016 0.6714021 0.8462108 0.8509803
5528 Lag 50 0.8985337 0.1983780 0.6138516 0.6233994
```

5529 Whichever function you use, if you find that your supposedly long Markov chain
5530 has not generated enough pseudo-iid samples, you should consider a longer run. In
5531 the present case we see that autocorrelation is especially high for the parameter
5532 `sigma` and our effective sample size for this parameter is 4!⁹ This means we would

⁹Anyone have any idea how the autocorrelation in `sigma` could be reduced?

have to run the model for much longer to obtain a reasonable effective sample size. Unfortunately, with many SCR models we observe high degrees of serial autocorrelation, which means we have to run long chains to obtain enough samples that can be considered iid, in order to obtain reasonable estimates of our parameters and their variances. What exactly constitutes a reasonable effective sample size is hard to say, but as a rule of thumb you should probably aim at several hundreds of these pseudo-iid samples. A more meaningful 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 which we introduced in Chapter 2. The MC error decreases with increasing sample size and its magnitude can thus be controlled by adjusting the length of the Markov chain. As a rule of thumb, the MC error should be 1% or less of the parameter estimate. Once you have reached this level, the estimates of the mean, standard error and 95% quantiles should no longer change significantly with additional iterations. For highly correlated samples, it will take more iterations to reduce the MC error. In coda, the MC error is given as part of the summary results (see below). Another option to deal with the serial autocorrelation of samples is to 'thin' Markov chains by some rate r and save only every r -th iteration. But as discussed in Chapter 2, this is not efficient and should only be applied if needed for practical reasons (e.g. a large number of parameters and iterations may force you to thin your samples so you object storing the model output does not become unmanageably large). For now, let's continue using this small set of samples to continue looking at the output.

11.6.5 Summary results

Now that we checked that our chains apparently have converged and pretending that we have generated enough samples from the posterior distribution, we can look at the actual parameter estimates. The `summary()` function will return two sets of results: the mean parameter estimates, with their standard deviation, the naive standard error - i.e. your regular standard error calculated for K (= number of iterations) samples without accounting for serial autocorrelation - and the corrected MC error (Time-series SE), which accounts for autocorrelation. In WinBUGS, this latter value is referred to as MC error and is only given in the log output within BUGS itself. You should adjust the `summary()` call by removing the burn-in from calculating parameter summary statistics. To do so, use the `window()` command, which lets you specify at which iteration to start 'counting'. In contrast to WinBUGS, which requires you to set the burn-in length before you run the model, this command gives us full flexibility to make decisions about the burn-in after we have seen the trajectories of our Markov chains. For our example, `summary(window(chain, start=1001))` returns the following output:

```
Iterations = 1001:5000
Thinning interval = 1
```



```

5573 Number of chains = 1
5574 Sample size per chain = 4000
5575
5576 1. Empirical mean and standard deviation for each variable,
5577    plus standard error of the mean:

```

```

5578
5579      Mean      SD Naive SE Time-series SE
5580 sigma  1.9986  0.13805 0.0021827      0.016091
5581 lam0   0.1096  0.01523 0.0002407      0.001401
5582 psi    0.6113  0.09148 0.0014465      0.010734
5583 N      489.8535 71.79695 1.1352094      8.431119

```

```

5584
5585 2. Quantiles for each variable:

```

```

5586
5587      2.5%      25%      50%      75%      97.5%
5588 sigma  1.75780  1.89847  1.9900  2.0944  2.2772
5589 lam0   0.08357  0.09824  0.1087  0.1192  0.1427
5590 psi    0.45110  0.54838  0.6052  0.6639  0.8192
5591 N      366.00000 440.00000 485.0000 530.0000 654.0000

```

5592 Looking at the MC errors, we see that in spite of the high autocorrelation, the
 5593 MC error for sigma is below the 1Our algorithm gives us a posterior distribution of
 5594 N, but we are usually interested in the density, D. Density itself is not a parameter
 5595 of our model, but we can derive a posterior distribution for D by dividing each
 5596 value of N (N at each iteration) by the area of the state-space (here 3032.719 km²)
 5597 and we can use summary statistics of this distribution to characterize D:

```

5598 > summary(window(chain[,4]/ 3032.719, start=1001))
5599 Iterations = 1001:5000
5600 Thinning interval = 1
5601 Number of chains = 1
5602 Sample size per chain = 4000
5603
5604 1. Empirical mean and standard deviation for each variable,
5605    plus standard error of the mean:
5606
5607      Mean      SD      Naive SE Time-series SE
5608 0.1615229 0.0236741 0.0003743      0.0027801
5609
5610 2. Quantiles for each variable:
5611
5612      2.5%      25%      50%      75%      97.5%
5613 0.1207 0.1451 0.1599 0.1748 0.2156

```

If we compare our mean density of 0.16/km² (and other parameters) with results from the same model run in secr and WinBUGS in Chapter XX, we see that estimates are almost identical (Table 1).

11.6.6 Other useful commands

While inspecting the time series plot gives you a first idea of how well you tuned your MH algorithm, use `rejectionRate()` to obtain the rejection rates (1 - acceptance rates) of the parameters that are written to your output:

```
> rejectionRate(chain)
      sigma      lam0      psi      N
0.44108822 0.77675535 0.00000000 0.01940388
```

Recall that rejection rates should lie between 0.2 and 0.8, so our tuning seems to have been appropriate here. Psi is never rejected since we update it with Gibbs sampling, where all candidate values are kept. And since N is the sum of all z, all it takes for N to change from one iteration to the next are small changes in the z-vector, so the rejection rate of N is always low. If you have run several parallel chains, you can combine them into a single mcmc object using the `mcmc.list()` command on the individual chains (note that each chain has to be converted to an mcmc object before combining them with `mcmc.list()`). You can then easily obtain the Gelman-Rubin diagnostic (Gelman et al., 2004), in WinBUGS called R-hat, using `gelman.diag()`, which will indicate if all chains have converged to the same stationary distribution. For details on these and other functions, see the coda manual, which can be found together with the package on the CRAN mirror.

11.7 MANIPULATING THE STATE-SPACE

So far, we have constrained the location of the activity centers to fall within the outermost coordinates of our rectangular state space by posing upper and lower bounds for x and y. But what if S has an irregular shape maybe there is a large water body we would like to remove from S, because we know our terrestrial study species does not occur there. Or the study takes place in a clearly defined area such as an island. As mentioned before, this situation is difficult to handle in WinBUGS. In some simple cases we can adjust the state space by setting `SXi` to be some function of `SYi` or vice versa. In this manner, we can cut off corners of the rectangle to approximate the actual state space. In R, we are much more flexible, as we can use the actual state-space polygon to constrain out si.¹⁰ To illustrate that, let's look at a camera trapping study of Florida panthers (*Puma concolor coryi*) conducted in the Picayune Strand Restoration Project (PSRP) area, southwest Florida (Fig. 7), by XXX, and financed by XXX. In the 1960ies the PSRP area was slated for

¹⁰ Have to check if we can use panther stuff for the book; otherwise, use raccoon example.

housing development, but then bought back by the State of Florida and is currently being restored to its original hydrology and vegetation. In an effort to estimate the density of the local Florida panther population, 98 camera traps were operated in the area for 21 months between 2005 and 2007. Florida panthers are wide-ranging animals and in order to account for their wide movements, the state-space was defined as the trapping grid buffered by 15 km around its outermost coordinates. However, the resulting rectangle contained some ocean in its southwestern corner (Fig. 7). In order to precisely describe the state-space, the ocean has to be removed. You can create a precise state-space polygon in ArcGIS and read it into R, or create the polygon directly within R. In the present case we intersected two shape files one of the state of Florida and one of the rectangle defined by a strip of 15 km around the camera-trapping grid. While you will most likely have to obtain the shapefile describing the landscape of and around your trapping grid (coastlines, water bodies etc.) from some external source, a polygon shapefile buffering your outermost trapping grid coordinates can easily be written in R.

If `xmin`, `xmax`, `ymin` and `ymax`, mark the outermost `x` and `y` coordinates of your trapping grid and `b` is the distance you want to buffer with, load the package `shapefiles` (Stabler, 2006) and use:

```

5667 x1= xmin-b
5668 xu= xmax+b
5669 y1= ymin-b
5670 yu= ymax+b
5671
5672 dd <- data.frame(Id=c(1,1,1,1,1),X=c(x1,xu,xu,x1,x1),Y=c(y1,y1,yu,yu,y1)) #create data fra
5673 ddTable <- data.frame(Id=c(1),Name=c("Item1"))
5674 ddShapefile <- convert.to.shapefile(dd, ddTable, "Id", 5) #convert #to shapefile, type pol
5675 write.shapefile(ddShapefile, 'c:/', arcgis=T) # save to location of #choice

```

You can read shapefiles into R loading the package `maptools` (Lewin-Koh et al., 2011) and using the function `readShapeSpatial()`. Make sure you read in shapefiles in UTM format, so that units of the trap array, the movement parameter `sigma` and the state-space are all identical. Intersection of polygons can be done in R also, using the package `rgeos` (Bivand and Rundel, 2011) and the function `gIntersect()`. The area of your single - polygon can be extracted directly from the state-space object `SSp`:

```

5683 > area <- SSp@polygons[[1]]@Polygons[[1]]@area /1000000

```

Note that dividing by 1000000 will return the area in km² if your coordinates describing the polygon are in UTM. If your state-space consists of several disjunct polygons, you will have to sum the areas of all polygons to obtain the size of the state-space. To include this polygon into our MCMC sampler we need one last spatial R package `sp` (Pebesma and Bivand, 2011), which has a function, `over()`,

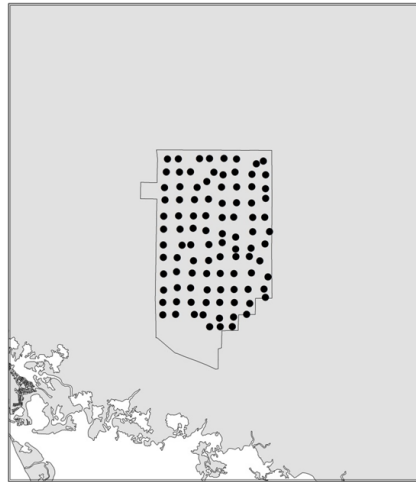


Figure 11.7. Rectangular state-space for a Florida panther camera trapping study in the PSRP area (grey outline, red block inset map of Florida) contain some ocean (white) that needs to be removed from the state-space.

```

5689 which allows us to check if a pair of coordinates falls within a polygon or not. All
5690 we have to do is embed this new check into the updating steps for s:

5691         Scand <- as.matrix(cbind(rnorm(M, S[,1], 2),
5692                                 rnorm(M, S[,2], 2)))          #draw candidate value
5693
5694 Scoord<-SpatialPoints(Scand*1000)      #convert to spatial points on UTM (m) scale
5695 SinPoly<-over(Scoord,SSp) # check if scand is within the polygon
5696
5697         for(i in 1:M) {
5698 if(is.na(SinPoly[i])==FALSE) { #if scand falls within polygon, continue update
5699   [rest of the updating step remains the same]

```

```

5700 Note that it is much more time-efficient to draw all M candidate values for s and
5701 check once if they fall within the state-space, rather than running the over() com-
5702 mand for every individual pair of coordinates. To make sure that our initial values
5703 for s also fall within the polygon of S, we use the function runifpoint() from the
5704 package spatstat (Baddeley and Turner, 2005), which generates random uniform
5705 points within a specified polygon. You'll find this modified MCMC algorithm in
5706 the online supplementary material (SCR0poisSSp). Finally, observe that we are
5707 converting candidate coordinates of S back to meters to match the UTM polygon.
5708 In all previous examples, for both the trap locations and the activity centers we

```

5709 have used UTM coordinates divided by 1000 to estimate sigma on a km scale. This
5710 is adequate for wide ranging individuals like bears. In other cases you may center
5711 all coordinates on 0. No matter what kind of transformation you use on your co-
5712 ordinates , make sure to always convert candidate values for S back to the original
5713 scale (UTM) before running the over() command.

11.8 MCMC SOFTWARE PACKAGES

5714 Throughout most of this book we will use WinBUGS and, occasionally, JAGS to
5715 run MCMC analyses. Here, we will briefly discuss the main pros and cons of these
5716 two programs as well as WinBUGS successor OpenBUGS. You can find scripts to
5717 simulate data and run the basic SCR model in all three programs in the online
5718 supplementary material (simSCR0poisBUGS).

11.8.1 WinBUGS

5720 In a nutshell, WinBUGS (and the other programs) do everything that we just went
5721 through in this chapter (and quite a bit more). Looking through your model, Win-
5722 BUGS determines which parameters it can use standard Gibbs sampling for (i.e.
5723 for conjugate full conditional distributions). Then, it determines, in the following
5724 hierarchy, whether to use adaptive rejection sampling, slice sampling or in the
5725 'worst' case Metropolis-Hastings sampling for the other full conditionals (Spiegel-
5726 halter et al., 2003). If it uses MH sampling, it will automatically tune the updater
5727 so that it works efficiently. While WinBUGS is a convenient piece of software that
5728 is still widely used, its major drawback is that it is no longer being developed, i.e.
5729 no new functions or distributions are added and no bugs are fixed.

11.8.2 OpenBUGS

5731 OpenBUGS is essentially the successor of WinBUGS. While the latter is no longer
5732 worked on, OpenBUGS is constantly developed further. The name 'OpenBUGS'
5733 refers to the software being open source, so users do not need to download a license
5734 key, like they have to for WinBUGS (although the license key for WinBUGS is free
5735 and valid for life).

5736 Compared to WinBUGS, OpenBUGS has a lot more built-in functions. The
5737 method of how to determine the right updater for each model parameter has
5738 changed and the user can manually control the MCMC algorithm used to update
5739 model parameters. Several other changes have been implemented in OpenBUGS
5740 and a detailed list of differences between the two BUGS versions, can be found at
5741 <http://www.openbugs.info/w/OpenVsWin>

5742 While OpenBUGS is a useful program for a lot of MCMC sampling applications,
5743 for reasons we do not understand, simple SCR models do not converge in Open-
5744 BUGS. It is therefore advisable that you check any OpenBUGS SCR model results

against result from WinBUGS. Also, currently, the R package BRugs (Thomas et al., 2006) necessary for running OpenBUGS through R has problems with 64-bit machines, so you may have to use the 32-bit version of R and OpenBUGS in order to make it work. The BUGS project site at <http://www.openbugs.info> provides a lot of information on and download links for OpenBUGS.

There is an extensive help archive for both WinBUGS and OpenBUGS and you can subscribe to a mailing list, where people pose and answer questions of how to use these programs at <http://www.mrc-bsu.cam.ac.uk/bugs/overview/list.shtml>

11.8.3 JAGS Just Another Gibbs Sampler

JAGS, currently at Version 3.1.0, is another free program for analysis of Bayesian hierarchical models using MCMC simulation. Originally, JAGS was the only program using the BUGS language that would run on operating systems other than the 32 bit Windows platforms. By now, there are OpenBUGS versions for Linux or Macintosh machines. JAGS 'only' generates samples from the posterior distribution; analysis of the output is done in R either by running JAGS through R using either the packages `rjags` (Plummer, 2011) or `R2jags` (Su and Yajima, 2011), or by using `coda` on your JAGS output. The program, manuals and `rjags` can be downloaded at <http://sourceforge.net/projects/mcmc-jags/files/> When run from within R using the package `rjags` or `R2jags`, writing a JAGS model is virtually identical to writing a WinBUGS model. However, some functions may have slightly different names and you can look up available functions and their use in the JAGS manual. One potential downside is that JAGS can be very particular when it comes to initial values. These may have to be set as close to truth as possible for the model to start. Although JAGS lets you run several parallel Markov chains, this characteristic interferes with the idea of using overdispersed initial values for the different chains. Also, we have occasionally experienced JAGS to crash and take the R GUI with it. Only re-installing both JAGS and `rjags` seemed to solve this problem. On the plus side, JAGS usually runs a little faster than WinBUGS, sometimes considerably faster (see section 4.XYZ), is constantly being developed and improved and it has a variety of functions that are not available in WinBUGS. For example, JAGS allows you to supply observed data for some deterministic functions of unobserved variables. In BUGS we cannot supply data to logical nodes. Another useful feature is that the adaptive phase of the model (the burn-in) is run separately from the sampling from the stationary Markov chains. This allows you to easily add more iterations to the adaptive phase if necessary without the need to start from 0. There are other, more subtle differences and there is an entire manual section on differences between JAGS and OpenBUGS. For questions and problems there is a JAGS forum online at <http://sourceforge.net/projects/mcmc-jags/forums/forum/610037>.

¹¹

¹¹As we make progress on the book, let's be sure to add linkages to places where we use JAGS in examples.

11.9 SUMMARY AND OUTLOOK

While there are a number of flexible and extremely useful software packages to perform MCMC simulations, it sometimes is more efficient to develop your own MCMC algorithm. Building an MCMC code follows three basic steps: Identify your model including priors and express full conditional distributions for each model parameter. If full conditionals are parametric distributions, use Gibbs sampling to draw candidate parameter values from this distributions; otherwise use Metropolis-Hastings sampling to draw candidate values from a proposal distribution and accept or reject them based on their posterior probability densities. These custom-made MCMC algorithms give you more modeling flexibility than existing software packages, especially when it comes to handling the state-space: In BUGS (and JAGS for that matter) we define a continuous rectangular state-space using the corner coordinates to constrain the Uniform priors on the activity centers s . But what if a continuous rectangle isn't an adequate description of the state-space? In this chapter we saw that in R it only takes a few lines of code to use any arbitrary polygon shapefile as the state-space, which is especially useful when you are dealing with coastlines or large bodies of water that need removing from the state-space. Another example is the SCR R package SPACECAP (Gopalaswamy et al., 2011) that was developed because implementation of an SCR model with a discrete state-space was inefficient in WinBUGS. Another situations in which using BUGS/JAGS becomes increasingly complicated or inefficient is when using point processes other than the Uniform Poisson point process which underlies the basic SCR model (see Chapter X). In the Chapters 9 and XX you will see examples of different point processes, implemented using custom-made MCMC algorithms.¹² Finally, the Chapters XX and XX deal with unmarked or partially marked populations using hand-made MCMC algorithms to handle the (partially) latent individual encounter histories. While some of these models can be written in BUGS/JAGS,¹³ they are painstakingly slow; others cannot be implemented in BUGS/JAGS at all. In conclusion, while you can certainly get by using BUGS/JAGS for standard SCR models, knowing how to write your own MCMC sampler allows you to tailor these models to your specific needs.

¹²Richard, Beth expand on that?

¹³the Poisson one for partially marked we wrote in BUGS and it should work with a known number of marked; the Bernoulli in JAGS with the `dsum()` function should work for the fully unknown; maybe some others? I don't remember. We may have to try writing the others before saying that they don't work in BUGS/JAGS; they are certainly much faster in R, though.

12

5814

5815

5816

GOODNESS OF FIT AND STUFF

5817
5818
5819

13

COVARIATE MODELS

STATE-SPACE COVARIATES

Underlying all spatial capture recapture models is a point process model describing the distribution of individual activity centers (\mathbf{s}_i) within the state space (\mathcal{S}). So far we have focused our discussion on the homogeneous binomial point process, $\mathbf{s}_i \sim \text{Uniform}(\mathcal{S}), i = 1, 2, \dots, N$, where N is the size of the population. This is a model of “spatial-randomness”¹ because the intensity of the activity centers is constant across the study area and the activity centers are distributed independently of each other.

The spatial-randomness assumption is often viewed as restrictive because ecological processes such as territoriality and habitat selection can result in non-random distributions of organisms. We have argued, however, that this assumption is less restrictive than may be recognized because the homogeneous point process actually allows for infinite possible configurations of activity centers. Furthermore, given enough data, the uniform prior will have very little influence on the estimated locations of activity centers. Nonetheless, the homogeneous point process model does not allow one to model population density using covariates—a central objective of much ecological research. For example, a homogeneous point process model may result in a density surface map indicating that individuals were more abundant in one habitat than another, but it does not do so explicitly. A more direct approach would be to model density using covariates as is done in generalized linear models (GLMs).

In this chapter we will present a method for fitting inhomogeneous binomial point process models using covariates in much the same way as is done with GLMs. The covariates we consider differ from those covered in previous chapters, which were typically attributes of the animal (*e.g.* sex, age) and were used to model movement or encounter rate. In contrast, here we wish to model covariates that

¹The phrase “complete spatial-randomness” is reserved for the homogeneous Poisson point process

are defined for all points in \mathcal{S} , which we will refer to as state-space, or density, covariates. These may include continuous covariates such as elevation, or discrete covariates such as habitat type.

Borchers and Efford (2008) were the first to propose an inhomogeneous point process model for SCR models, and our approach is similar to theirs with the exception that we will use a binomial rather than a Poisson model because the binomial model is easily integrated into our data augmentation scheme and is consistent with the objective of determining how a *fixed* number of activity centers are distributed with respect to covariates.

The method we use to accommodate inhomogeneous binomial point process models within our MCMC algorithm is simple—we replace the uniform prior with a prior describing the distribution of the N activity centers conditional on the covariates. Development of this prior, which does not have a standard form, is a central component of this chapter. First we will begin with a review of homogeneous point process models.

14.1 HOMOGENEOUS POINT PROCESS REVISITED

The homogeneous Poisson point process is *the* model of “complete spatial randomness” and is often used in ecology as a null model to test for departures from randomness. Given its central role in the analysis of point processes, it is helpful to compare it with the binomial model that we use in our SCR models. The primary descriptor of the homogeneous point process model is the “intensity” parameter, μ which describes the expected number of points in an infinitesimally small area. The intensity parameter can also be used to determine the expected number of points in any region of the state-space \mathcal{S} . To denote this, we say that the expected number of points in region $B \in \mathcal{S}$ is $n(B) = A(B)\mu$ where $A(B)$ is the area of region B . In words, the expected number of points in B is simply the area of B multiplied by the intensity parameter. One property of the Poisson model is that if we divide the entire state-space into $k = 1, \dots, K$ disjunct regions, the counts $\mathbf{n}(\mathbf{B})$ are independent and identically distributed, (*i.i.d.*). This is one of the distinctions between the Poisson model and the binomial model, for which the counts $n(B_k)$ are not *i.i.d.* as we will explain shortly. This difference is also related to another distinction between the two models, namely that the binomial model conditions on the number of points to be simulated N ; whereas under the Poisson model N is random. Here is some simple **R** code to illustrate this point.

```

5881 mu <- 4                                # intensity
5882 Np <- rpois(1, mu)                    # Np is random
5883 PPP <- cbind(runif(Np), runif(Np))    # Poisson point process
5884
5885 Nb <- 4                                # Nb is fixed
5886 BPP <- cbind(runif(Nb), runif(Nb))    # Binomial point process

```

5887 Note that in both models, the N points are independent of one another and
 5888 distributed uniformly throughout \mathcal{S} . Thus, the intensity at any point $x \in \mathcal{S}$ is
 5889 $\mu = 1/A(\mathcal{S})$ where $A(\mathcal{S})$ denotes the area of the state-space. In the **R** code above,
 5890 the area of the state-space is 1 unit, and thus the intensity is $\mu = 1/1$.

5891 Although the Poisson model is typically described in terms of μ , the binomial
 5892 model is not; rather, it is more common to consider a discrete state space, such
 5893 as a grid with K pixels. Under the binomial model, the number of points
 5894 in each region is $n(B_k) \sim \text{Bin}(N, p_k)$ where $p_k = A(B_k)/A(\mathcal{S})$, ie p_k is simply the
 5895 fraction of the state-space area in B_k . This discrete space representation of the
 5896 binomial point process is shown in Fig. 14.1. The state-space in this case is the
 5897 unit square, and thus the probability of a point falling in each of the 25 disjunct
 5898 regions is $p_k = 1/25$ and thus the expected counts are simply $\mathbb{E}(n(B_k)) = Np_k$. In
 5899 the figure $N = 50$ and thus we would expect 2 points per pixel, which happens to be
 5900 the empirical mean of the data in Fig. 14.1. Note also that these counts are not in-
 5901 dependent realizations from a binomial distribution since $\sum_k n(B_k) = N$. Instead,
 5902 the model for the entire vector is $\mathbf{n}(\mathbf{B}) \sim \text{Multinomial}(N, \pi = (p_1, p_2, \dots, p_K))$
 5903 (Illian, 2008b). The dependence among counts has virtually no practical conse-
 5904 quence when the number of pixels is large. For example, if we have 100 pixels,
 5905 the number of counts in one pixels tells you very little about the expected count
 5906 in another pixel. However, if there are only 2 pixels, then clearly the number of
 5907 points in one pixel tells you exactly how many will occur in the remaining pixel. To
 5908 gain familiarity with the multinomial distribution and the discrete representation of
 5909 space, use the `rmultinom` function in **R** to simulate counts similar to those shown
 5910 in Fig. 14.1, for example using a command such as:

```
5911 n.B_k <- rmultinom(1, size=50, prob=rep(1/25, 25))
5912 matrix(n.B_k, 5, 5)
```

5913 The discrete space representation of the binomial point process is of practical
 5914 importance when fitting SCR models because spatial covariates are almost always
 5915 represented in a discrete format, often called “rasters” in GIS-speak. In such cases,
 5916 we often need to change our definition of the prior for an activity center from
 5917 $s_i \sim \text{Uniform}(\mathcal{S})$ to $s_i \sim \text{Multinomial}(1, \pi)$. In the latter case, the activity
 5918 center is simply defined as an integer representing pixel “id”. Note also that the
 5919 multinomial distribution with an index of 1 (*i.e.* `size=1` in `rmultinom`) is referred
 5920 to as the categorical distribution, which we will frequently use in the BUGS language.

14.2 INHOMOGENEOUS BINOMIAL POINT PROCESS

5921 As with the homogeneous model, the inhomogeneous binomial point process model
 5922 is developed conditional on N . The primary distinction is that the uniform distri-
 5923 bution is replaced with another distribution allowing for the intensity parameter to
 5924 vary spatially. To arrive at this new distribution, define $\mu(x, \beta)$ to be a function of

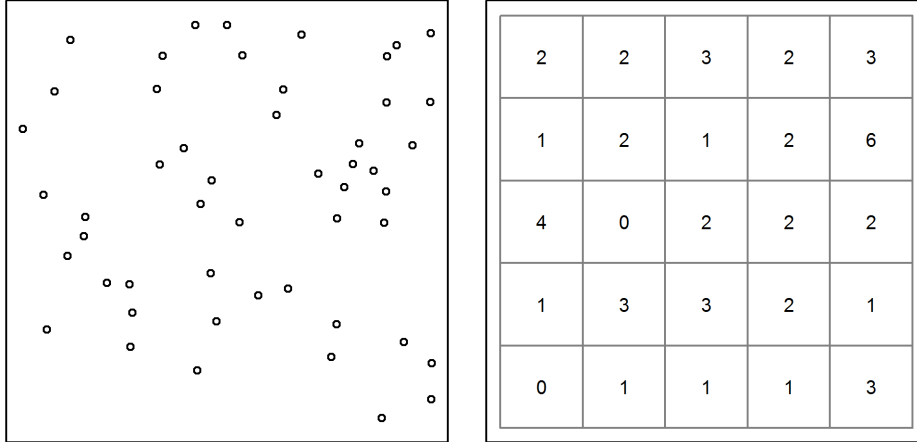


Figure 14.1. Homogeneous binomial point process with $N=50$ points represented in continuous and discrete space.

spatially-referenced covariates (β) available at all points of the state space. To be
concise we will subsequently drop the vector of coefficients from our notation, and
simply use $\mu(x)$. Since an intensity must be strictly positive, it is natural to model
 $\mu(x)$ using the log-link.

$$\log(\mu(x)) = \sum_{j=1}^J \beta_j v_j(x), \quad x \in \mathcal{S}$$

where β_j is the regression coefficient for covariate $v_j(x)$. To be clear, $v(x)$ is the
value of any covariate, such as habitat type or elevation, at location x . This equa-
tion should look familiar because it is the standard linear model used in log-linear
GLMs. Note, however, that we have no need for an intercept because it would be
confounded with N . This should be intuitive since an intercept would represent the
expected value of N when $\beta = 0$, but we already have a parameter in the model for
expected abundance, namely $\mathbb{E}[N] = \psi M$. Thus an intercept would be redundant,
and without it we are still able to achieve our goal of describing the distribution of
 N activity centers as a function of spatial covariates.

Now that we have a model of the intensity parameter $\mu(x)$, we need to develop
the associated probability density function to use in place of the uniform prior.
Remembering that the integral of a pdf must be unity, we can create a pdf by
dividing $\mu(x)$ by a normalizing constant, which in this case is the integral of $\mu(x)$
evaluated over the entire state-space. **ANDY, is there a better justification for this?**

5943 The probability density function is therefore

$$f(x) = \frac{\mu(x)}{\int_{x \in S} \mu(x) dx} \quad (14.2.1)$$

5944 Substituting this distribution for the uniform prior allows us to fit inhomogeneous
 5945 binomial point process models to spatial capture-recapture data. We can also use
 5946 this distribution to obtain the expected number of individuals in any given region.
 5947 Specifically, the proportion of N expected to occur in any region B when hetero-
 5948 geneity in density is present is $p(B) = \int_B f(x) dx$. These are also the multinomial
 5949 cell probabilities if the regions are disjoint and compose the entire state-space.

5950 As a practical matter, note that the integral in the denominator of $f(x)$ is
 5951 evaluated over space, and since we almost always regard space as two-dimensional,
 5952 this is a two-dimensional integral that can be approximated using the methods
 5953 discussed in refChXXX. These methods include Monte Carlo integration, Gaussian
 5954 quadrature, etc... Alternatively, if our state-space covariates are in raster format,
 5955 *i.e* they are in discrete space, the integral can be replaced with a sum over all pixels,
 5956 which is much more efficient computationally.

5957 We now have all the tools needed to fit inhomogeneous point process (IPP)
 5958 models. Before doing so, we note that the IPP for the activity centers results in
 5959 another IPP for the observation process, $\lambda(x)$. As a reminder, $\lambda(x)$ is the expected
 5960 number of captures for a trap at point x . As was true for the homogeneous model,
 5961 this intensity function is a product of the point process intensity and the encounter
 5962 rate function, $\lambda(x) = \mu(x)g(x)$.

5963 In the next section we walk through a few examples, building up from the
 5964 simplest case where we actually observe the activity centers as though they were
 5965 data. In the second example, we fit our new model to simulated data in which
 5966 density is a function of a single continuous covariate. Example three shows an
 5967 analysis in discrete space using both **secr** (Efford, 2011) and **JAGS** (Plummer,
 5968 2003). In the last example, we model the intensity of activity centers for a real
 5969 dataset collected on jaguars (*Panthera onca*) in Argentina.

14.3 EXAMPLES

5970 14.3.1 Simulation and analysis of inhomogeneous point processes

5971 In SCR models, the point process is not directly observed, but in other contexts
 5972 it is. Examples include the locations of disease outbreaks or the locations of trees
 5973 in a forest. Fitting inhomogeneous point process models to such data is straight-
 5974 forward and illustrates the fundamental process that we will later embed in our
 5975 MCMC algorithm used to fit SCR models.

5976 Suppose we knew the locations of 100 animals' activity centers. To estimate
 5977 the intensity surface $\mu(x)$ underlying these points, we need to derive the likelihood
 5978 for our data under this model. Given the pdf $f(x)$ (Eq. 14.2.1) and assuming that

the points are mutually independent of one another, we may write the likelihood as the product of R such terms, where $R = 100$ is the sample size in this case, *ie* the observed number of activity centers.

$$\mathcal{L}(\beta|\mathbf{x}_i) = \prod_{i=1}^R f(x_i)$$

Having defined the likelihood we could choose a prior and obtain the posterior for β using Bayesian methods, or we can find the maximum likelihood estimates (MLEs) using standard numerical methods as is demonstrated below.

First, let's simulate some data. Simulating data under an inhomogeneous point process model is often accomplished using indirect methods such as rejection sampling. Rejection sampling proceeds by simulating data from a standard distribution and then accepting or rejecting each sample using probabilities defined by the distribution of interest. For more information, readers should consult an accessible text such as Robert and Casella (2004). In our example, we simulate from a uniform distribution and then accept or reject using the (scaled) probability density function $f(x)$. Note that we first define a spatial covariate (elevation) that is a simple function of the spatial coordinates increasing from the southwest to the northeast of our state-space.²

The following **R** commands demonstrate the use of rejection sampling to simulate an inhomogeneous point process for the covariate depicted in Fig. 14.3.1. The code uses the **cuhre** function in the **R2Cuba** package to integrate the intensity function over space (Hahn et al., 2011).

```
# spatial covariate (with mean 0)
elev.fn <- function(x) x[1]+x[2]-1
# intensity function
mu <- function(x, beta) exp(beta*elev.fn(x=x))

# Simulate PP using rejection sampling
set.seed(300225)
N <- 100
count <- 1
s <- matrix(NA, N, 2)
beta <- 2 # parameter of interest
int.mu <- R2Cuba::cuhre(2, 1, mu, beta=beta)$value
elev.min <- elev.fn(c(0,0)) #elev.fn(cbind(0,0))
elev.max <- elev.fn(c(1,1)) #elev.fn(cbind(1,1))
Q <- max(c(exp(beta*elev.min) / int.mu, #2d(beta),
           exp(beta*elev.max) / int.mu)) #2d(beta))
while(count <= 100) {
```

²Such functional forms of covariates are rarely available, which is why continuous spatial covariates are more often measured on a discrete grid.

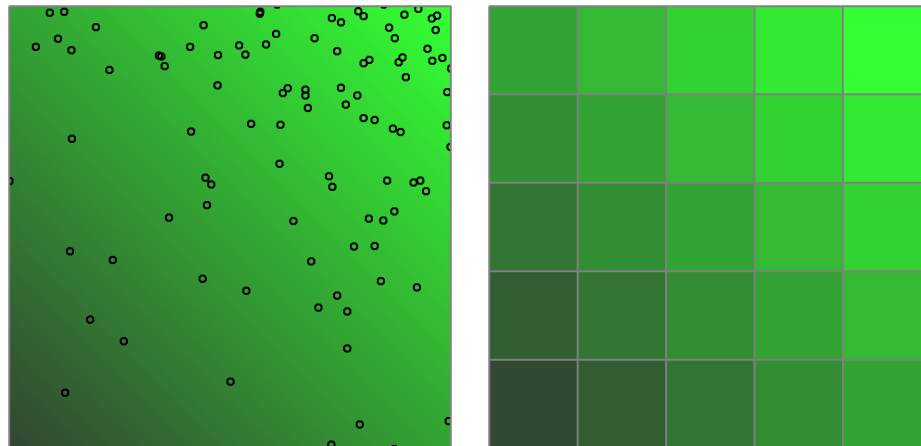


Figure 14.2. An example of a spatial covariate, say elevation, and a realization of an inhomogeneous binomial point process with $N=100$ and $\mu(x) = \exp(\beta \text{Elev})$ where $\beta = 2$.

```

6016 x.c <- runif(1, 0, 1); y.c <- runif(1, 0, 1)
6017 s.cand <- c(x.c,y.c)
6018 pr <- exp(beta*elev.fn(s.cand)) / int.mu #2d(beta)
6019 if(runif(1) < pr/Q) {
6020   s[count,] <- s.cand
6021   count <- count+1
6022 }
6023 }

```

6024 The simulated data are shown in Fig 14.3.1. High elevations are represented by
 6025 light green and low elevations by dark green. The activity centers of one hundred
 6026 animals are shown as points, and it is clear that these simulated animals prefer the
 6027 high elevations. Perhaps they are mountain goats. The underlying model describing
 6028 this preference is $\log(\mu(x)) = \exp(\beta \times \text{Elevation}(x))$ where $\beta = 2$ is the parameter
 6029 to be estimated and $\text{Elevation}(x)$ is a function of the coordinates at x , as displayed
 6030 on the map.

6031 Given these points, we will now estimate β by minimizing the negative-log-
 6032 likelihood using R's `optim` function.

```

6033 # Negative log-likelihood
6034 nll <- function(beta) {
6035   int.mu <- cuhre(2, 1, mu, beta=beta)$value
6036   -sum(beta*elev.fn(s) - log(int.mu))

```

```

6037 }
6038 starting.value <- 0
6039 fm <- optim(starting.value, nll, method="Brent",
6040             lower=-5, upper=5, hessian=TRUE)
6041 c(Est=fm$par, SE=sqrt(1/fm$hessian)) # estimates and SEs

```

6042 Maximizing the likelihood took a small fraction of a second, and we obtained
 6043 an estimate of $\hat{\beta} = 1.99$. We could plug in this estimate to our linear model at each
 6044 point in the state-space to obtain the MLE for the intensity surface.

6045 This example demonstrates that if we had the data we wish we had, *i.e.* if we
 6046 knew the coordinates of the activity centers, we could easily estimate the parameters
 6047 governing the underlying point process. Unfortunately, in SCR models, the activity
 6048 centers cannot be directly observed, but spatial re-captures, that is captures of
 6049 individuals at multiple locations in space, provide us with the information needed
 6050 to estimate these latent parameters.

6051 14.3.2 Fitting inhomogeneous point process SCR models

6052 Continuous space

6053 One of the nice things about hierarchical models is that they allow us to break a
 6054 problem up into a series of simple conditional relationships. Thus, we can simply
 6055 add the methods described above into our existing MCMC algorithm to simulate
 6056 the posteriors of β conditional on the simulated values of \mathbf{s}_i . To demonstrate, we
 6057 will continue with the previous example. Specifically, we will overlay a grid of
 6058 traps upon the map shown in Fig. 14.3.1. We will then simulate capture histories
 6059 conditional upon the activity centers shown on the map. Then, we will attempt to
 6060 estimate the activity center locations as though we did not know where they were,
 6061 as is the case in real applications.

6062 Here is some **R** code to simulate the encounter histories under a Poisson ob-
 6063 servation model, which would be appropriate if animals could be detected multiple
 6064 times at a trap during a single occasion.

```

6065 # Create trap locations
6066 xsp <- seq(-0.8, 0.8, by=0.2)
6067 len <- length(xsp)
6068 X <- cbind(rep(xsp, each=len), rep(xsp, times=len))
6069
6070 # Simulate capture histories, and augment the data
6071 ntraps <- nrow(X)
6072 T <- 5
6073 y <- array(NA, c(N, ntraps, T))
6074
6075 nz <- 50 # augmentation
6076 M <- nz+nrow(y)
6077 yz <- array(0, c(M, ntraps, T))

```

```

6078
6079 sigma <- 0.1 # half-normal scale parameter
6080 lam0 <- 0.5 # basal encounter rate
6081 lam <- matrix(NA, N, ntraps)
6082
6083 set.seed(5588)
6084 for(i in 1:N) {
6085   for(j in 1:ntraps) {
6086     distSq <- (s[i,1]-X[j,1])^2 + (s[i,2] - X[j,2])^2
6087     lam[i,j] <- exp(-distSq/(2*sigma^2)) * lam0
6088     y[i,j,] <- rpois(T, lam[i,j])
6089   }
6090 }
6091 yz[1:nrow(y),,] <- y # Fill

```

Now that we have a simulated capture-recapture dataset y , and we have augmented it to create the new data object yz , we are ready to begin sampling from the posteriors. A commented Gibbs sampler written in **R** is available in the accompanying **R** package **scrbook** (see ?scrIPP). There are two small parts of the **R** code that distinguish it from previous code we have shown to fit homogeneous point processes. First, we need to update the parameter β conditional on all other parameters in the model. The code to do so is:

```

6099 D1 <- cuhre(2, 1, mu, lower=c(xlims[1], ylims[1]),
6100             upper=c(xlims[2], ylims[2]), beta=beta1)$value
6101 beta1.cand <- rnorm(1, beta1, tune[3])
6102 D1.cand <- cuhre(2, 1, mu, lower=c(xlims[1], ylims[1]),
6103                 upper=c(xlims[2], ylims[2]), beta=beta1.cand)$value
6104 ll.beta1 <- sum( beta1*elev.fn.v(S) - log(D1) )
6105 ll.beta1.cand <- sum( beta1.cand*elev.fn.v(S) - log(D1.cand) )
6106 if(runif(1) < exp(ll.beta1.cand - ll.beta1) ) {
6107   beta1 <- beta1.cand
6108 }

```

Next, we need to put the new prior on the activity centers:

```

6110 #ln(prior), denominator is constant
6111 prior.S <- beta1*cov(S[i,1], S[i,2]) # - log(D1)
6112 prior.S.cand <- beta1*(Scand[1] + Scand[2]) # - log(D1)
6113 if(runif(1) < exp((ll.S.cand+prior.S.cand) - (ll.S+prior.S))) {
6114   S[i,] <- Scand
6115   lam <- lam.cand
6116   D[i,] <- dtmp
6117 }

```

We can apply this modified sampler to our data using the code shown in the help file for **scrIPP**. We obtain posterior distributions summarized in Table 14.2.

Mixing is good, and as usual, life is very nice when we are working with simulated data.

Fitting continuous space IPP models is somewhat difficult in **BUGS** because our prior $f(x)$ is not one of the available distributions that come with the software³ **secr** allows users to fit continuous space using polynomials of the x- and y- coordinates, but not for truly continuous covariates. However, these are not really important limitations because discrete space versions are straight-forward, and virtually all spatial covariates are defined as such.

Discrete space

To fit discrete space models, we follow the same steps as outlined in Chapter XXX—we define s_i as pixel ID, and we use the categorical distribution as a prior. A good example of this is in +citeKery capricaille. Here we present an analysis of the simulated data shown in the right panel of Fig. 14.3.1. The spatial covariate, let's call it elevation again, was simulated from a kriging type of model as shown on the help page `ch9simData` in `scrbook`. The points are the number of activity centers in each pixel, generated from a single realization of the IPP $mu(x) = 2elev$.

The **BUGS** code to fit an IPP model to these data is shown in the following panel.

```

model{
  sigma ~ dunif(0, 1)
  lam0 ~ dunif(0, 5)
  beta ~ dnorm(0,0.1)
  psi ~ dbeta(1,1)

  for(j in 1:nPix) {
    theta[j] <- exp(beta*elevation[j])
    probs[j] <- theta[j]/sum(theta[])
  }

  for(i in 1:M) {
    w[i] ~ dbern(psi)
    s[i] ~ dcat(probs[])
    x0g[i] <- Sgrid[s[i],1]
  }

```

³It is possible, if somewhat cumbersome, to add new distributions in **BUGS**.

Table 14.1. Posterior summaries from inhomogeneous point proces model

	Mean	SD	2.5%	50%	97.5%
$\sigma = 0.10$	0.1026	0.0048	0.0935	0.1025	0.1123
$\lambda_0 = 0.50$	0.4419	0.0493	0.3496	0.4400	0.5390
$\psi = 0.66$	0.6826	0.0554	0.5762	0.6820	0.7923
$\beta = 2.00$	2.1601	0.3390	1.5193	2.1583	2.8043
$N = 100$	102.7696	6.2689	92.0000	102.0000	117.0000

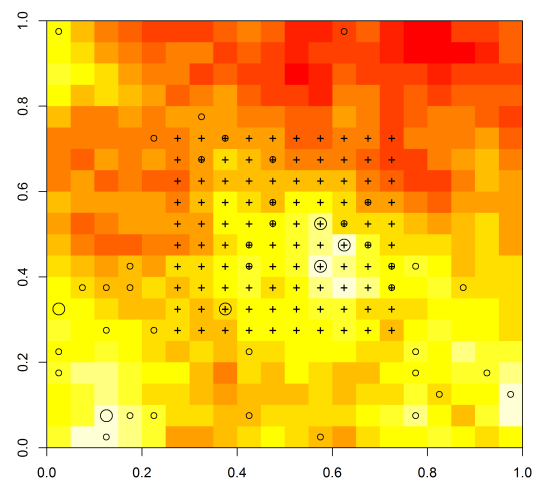


Figure 14.3. Simulated activity centers in discrete space. The spatial covariate, elevation, is highest in the higher areas. Density of activity centers (circles) increases with elevation. Trap locations are shown as crosses.

```

6153   y0g[i] <- Sgrid[s[i],2]
6154   for(j in 1:ntraps) {
6155     dist[i,j] <- sqrt(pow(x0g[i]-grid[j,1],2) + pow(y0g[i]-grid[j,2],2))
6156     lambda[i,j] <- lam0*exp(-dist[i,j]*dist[i,j]/(2*sigma*sigma)) * w[i]
6157     y[i,j] ~ dpois(lambda[i,j])
6158   }
6159 }
6160
6161 N <- sum(w[])
6162 Density <- N/1 # unit square
6163 }

```

6164 This model can also be fit in **secr**, which refers to the pixel locations as a
 6165 “mask”. **R** code to fit the models using **secr** and **JAGS** is available in **scrbook** ,
 6166 see **help(ch9secrYjags)**. Results of the comparison are shown in Table ?? and
 6167 are very similar as expected.

6168 Density surface maps can be created for fun, and of course to inform manage-
 6169 ment decisions. [describe how to do this]

6170 14.3.3 The jaguar data

6171 Estimating density of large felines has been a priority for many conservation orga-
 6172 nizations, but no robust methodologies existed before the advent of SCR. Distance
 6173 sampling is not feasible for such rare and cryptic species, and traditional capture-
 6174 recapture methods yield estimates that are highly sensitive to the subjective choice
 6175 of the effective survey area. In this example, we demonstrate how readily density
 6176 can be estimated for a globally imperilled species using SCR. Furthermore, we show
 6177 how inhomogeneous point process models can be used to test important hypotheses
 6178 regarding the factors affecting density.

6179 [describe study]

6180 A few aspects of this design are noteworthy. First, the dimensions and config-
 6181 uration of the trap array differed among the regions of the trap array. This fact
 6182 alone could explain variation in the number of animals exposed to sampling, which

Table 14.2. Comparison of **secr** and **JAGS** results

Software	Par	Est.	SD	lower	upper
secr	N	49.2803	5.7535	41.0087	64.3879
	β	2.1772	0.5628	1.0741	3.2804
	λ_0	0.9203	0.0764	0.7824	1.0825
	σ	0.0990	0.0038	0.0918	0.1068
JAGS	N	48.2072	5.4053	39.0000	60.0000
	β	2.1026	0.5323	1.0889	3.1506
	λ_0	0.9328	0.0766	0.7898	1.0921
	σ	0.1004	0.0041	0.0929	0.1089

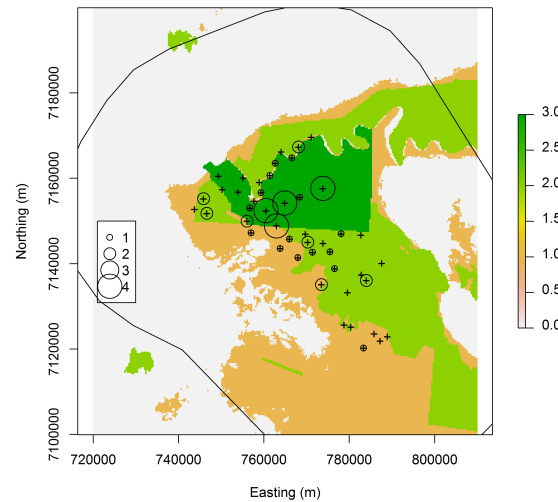


Figure 14.4. Jaguar detections

would have no biological meaning. Furthermore, the area of inference is an irregular polygon that was not sampled uniformly. Only by estimating density can we hope to extrapolate our estimates from the sampled region to get what we are after. In this case, this is readily accomplished since the entire state-space can be classified as one of the 3 levels of protection from poaching. Of course, it general it is always preferable to sample more uniformly throughout the area of interest in case some unmeasured covariate biases the extrapolation.

To assess the influence of poaching on jaguar density, we considered 2 metrics of poaching pressure, one political and one continuous measure of accessibility (Fig xxx).

14.4 SUMMARY

When state-space covariates are available, we can model density by replacing the uniform prior on the activity centers with a prior based on a normalized log-linear function of covariates. This yields a model of the inhomogeneous point process describing the location of activity centers, which can be used to test hypotheses about covariates affecting density. In rare cases, these covariates are truly continuous in the sense that they are defined as a function of space. More often, covariates are represented on rasters, which simplifies the analysis. Fitting these models can be accomplished using **BUGS**, **secr**, or the custom **R** code presented in this chapter

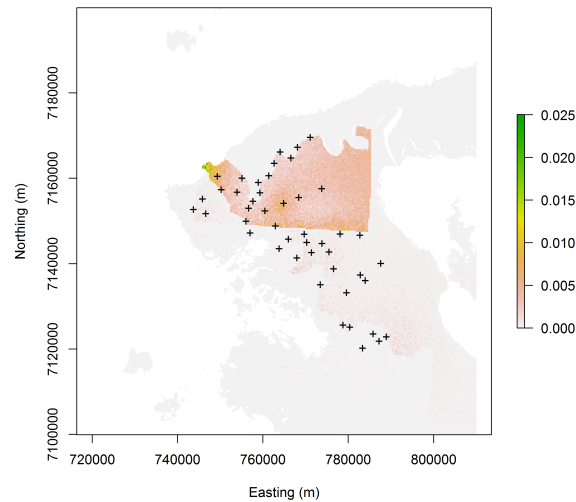


Figure 14.5. Estimated density surface for the jaguar dataset

6201 and found in the package `scrbook`.

6202 All the examples in this section included a single state-space covariate, but this
 6203 was for simplicity only. Including multiple covariates poses no additional challenges.
 6204 Likewise, additional model structure such sex-specific encounter rate parameters or
 6205 behavioral responses can be accommodated.

14.5 OTHER IDEAS

6206 Should have some discussion on some ideas for building flexible models. Might be
 6207 cool to use the Ickstadt/Wolpert as a model for the inhomogeneous point process.
 6208 Dont have to do it, just mention it. Also some kind of a spline model or similar.

15

6209

6210

6211

INHOMOGENEOUS POINT PROCESS

6212
6213
6214

16

OPEN MODELS

BIBLIOGRAPHY

- Alho, J. (1990), “Logistic regression in capture-recapture models,” *Biometrics*, 623–635.
- Arnason (1973), “Missing,” *Missing*, Missing, Missing.
- (1974), “Missing,” *Missing*, Missing, Missing.
- Baddeley, A. and Turner, R. (2005), “Spatstat: an R package for analyzing spatial point patterns,” *Journal of Statistical Software*, 12, 1–42, ISSN 1548-7660.
- Bales, S. L., Hellgren, E. C., Leslie Jr., D. M., and Hemphill Jr., J. (2005), “Dynamics of recolonizing populations of black bears in the Ouachita Mountains of Oklahoma,” *Wildlife Society Bulletin*, 1342–1351.
- Berger, J. O., Liseo, B., and Wolpert, R. L. (1999), “Integrated likelihood methods for eliminating nuisance parameters,” *Statistical Science*, 1–22.
- Bivand, R. and Rundel, C. (2011), *rgeos: Interface to Geometry Engine - Open Source (GEOS)*, r package version 0.1-8.
- Borchers, D. and Efford, M. (2008), “Spatially explicit maximum likelihood methods for capture-recapture studies,” *Biometrics*, 64, 377–385.
- Borchers, D. L. (missing), “missing,” *Missing*, missing.
- Borchers, D. L., Buckland, S. T., and Zucchini, W. (2002), *Estimating animal abundance: closed populations*, vol. 13, Springer Verlag.
- Boulanger, J. and McLellan, B. (2001), “Closure violation in DNA-based mark-recapture estimation of grizzly bear populations,” *Canadian Journal of Zoology*, 79, 642–651.
- Brooks, S. P., Catchpole, E. A., and Morgan, B. J. T. (2000), “Bayesian Animal Survival Estimation,” *Statistical Science*, 15, 357–376.
- Buckland, S. T. (2001), *Introduction to distance sampling: estimating abundance of biological populations*, Oxford, UK: Oxford University Press.
- Burnham, K. P. and Overton, W. S. (1978), “Estimation of the size of a closed population when capture probabilities vary among animals,” *Biometrika*, 65, 625.
- Casella, G. and George, E. I. (1992), “Explaining the Gibbs sampler,” *American Statistician*, 46, 167–174.
- Chandler, R. and Royle, J. (2012), “Spatially-explicit models for inference about density in unmarked populations,” *Biometrics (in review)*.
- Chandler, R., Royle, J., and King, D. (2011), “Inference about density and temporary emigration in unmarked populations,” *Ecology*, 92, 1429–1435.

- Converse, S. J. and Royle, J. A. (2010), “Missing,” *Missing*, Missing.
- Coull, B. A. and Agresti, A. (1999), “The Use of Mixed Logit Models to Reflect Heterogeneity in Capture-Recapture Studies,” *Biometrics*, 55, 294–301.
- Dawson, D. and Efford, M. (2009), “Bird population density estimated from acoustic signals,” *Journal of Applied Ecology*, 46, 1201–1209.
- Dice, L. R. (1938), “Some census methods for mammals,” *Journal of Wildlife Management*, 2, 119–130.
- Dorazio, R. and Royle, J. (2003), “Mixture models for estimating the size of a closed population when capture rates vary among individuals,” *Biometrics*, 351–364.
- Dorazio, R. M. (2007), “On the choice of statistical models for estimating occurrence and extinction from animal surveys,” *Ecology*, 88, 2773–2782.
- Durbin and Elston (2012), “Missing,” *Missing*, Missing.
- Efford, M. (2004), “Density estimation in live-trapping studies,” *Oikos*, 106, 598–610.
- (2011), “secre-spatially explicit capture-recapture in R,” .
- Efford, M., Dawson, D., and Robbins, C. (2004), “DENSITY: software for analysing capture-recapture data from passive detector arrays,” *Animal Biodiversity and Conservation*, 217–228.
- Fienberg, S. E., Johnson, M. S., and Junker, B. W. (1999), “Classical multilevel and Bayesian approaches to population size estimation using multiple lists,” *Journal of the Royal Statistical Society of London A*, 163, 383–405.
- Gardner, B. (2009), “missing,” *Missing*, missing.
- Gardner, B., Royle, J., Wegan, M., Rainbolt, R., and Curtis, P. (2010), “Estimating black bear density using DNA data from hair snares,” *The Journal of Wildlife Management*, 74, 318–325.
- Gelfand, A. and Smith, A. (1990), “Sampling-based approaches to calculating marginal densities,” *Journal of the American statistical association*, 85, 398–409.
- Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004), *Bayesian data analysis, second edition.*, Boca Raton, Florida, USA: CRC/Chapman & Hall.
- Gelman, A., Meng, X. L., and Stern, H. (1996), “Posterior predictive assessment of model fitness via realized discrepancies,” *Statistica Sinica*, 6, 733–759.
- Geman, S. and Geman, D. (1984), “Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6, 721–741.
- Genz, A. S., Meyer, M. R., Lumley, T., and Maechler, M. (2007), “The adapt Package. R package version 1.0-4,” .
- Gilks, W. and Wild, P. (1992), “Adaptive rejection sampling for Gibbs sampling,” *Applied Statistics*, 41, 337–348.
- Gilks, W. R., Thomas, A., and Spiegelhalter, D. J. (1994), “A Language and Program for Complex Bayesian Modelling,” *Journal of the Royal Statistical Society. Series D (The Statistician)*, 43, 169–177, ArticleType: primary_article / Issue Title: Special Issue: Conference on Practical Bayesian Statistics, 1992 (3) / Full publication date: 1994 / Copyright 1994 Royal Statistical Society.

- Gopalaswamy (2012), “Missing,” *Missing*, missing.
- Gopalaswamy, A. M., Royle, A. J., Hines, J., Singh, P., Jathanna, D., Kumar, N. S., and Karanth, K. U. (2011), *A Program to Estimate Animal Abundance and Density using Spatially-Explicit Capture-Recapture*, r package version 1.0.4.
- Hahn, T., Bouvier, A., and Kieu, K. (2011), “Package ‘R2Cuba’ R package version 1.0-6,” .
- Hastings, W. (1970), “Monte Carlo sampling methods using Markov chains and their applications,” *Biometrika*, 57, 97–109.
- Hawkins, C. and Racey, P. (2005), “Low population density of a tropical forest carnivore, *Cryptoprocta ferox*: implications for protected area management,” *Oryx*, 39, 35–43.
- Hestbeck (1991), “Missing,” *Missing*, Missing, Missing.
- Huggins, R. M. (1989), “On the statistical analysis of capture experiments,” *Biometrika*, 76, 133.
- Illian (2008a), “Missing,” *Missing*, Missing.
- Illian, J. (2008b), *Statistical analysis and modelling of spatial point patterns*, Wiley-Interscience.
- Ivan, J. (2012), “Density, demography, and seasonal movements of snowshoe hares in central Colorado,” Ph.D. thesis, COLORADO STATE UNIVERSITY.
- Jackson, R., Roe, J., Wangchuk, R., and Hunter, D. (2006), “Estimating Snow Leopard Population Abundance Using Photography and Capture-Recapture Techniques,” *Wildlife Society Bulletin*, 34, 772–781.
- Johnson (2010), “A Model-Based Approach for Making Ecological Inference from Distance Sampling Data,” *Biometrics*, 66, 310318.
- Johnson, D. (1999), “The insignificance of statistical significance testing,” *The journal of wildlife management*, 763–772.
- Karanth, K. U. (1995), “Estimating tiger *Panthera tigris* populations from camera-trap data using capture–recapture models,” *Biological Conservation*, 71, 333–338.
- Kendall (1997), “Missing,” *Missing*, missing.
- Kéry, M. (2010), *Introduction to WinBUGS for Ecologists: Bayesian Approach to Regression, ANOVA, Mixed Models and Related Analyses*, Academic Press.
- Kéry, M., Gardner, B., Stoeckle, T., Weber, D., and Royle, J. A. (2010), “Use of Spatial Capture-Recapture Modeling and DNA Data to Estimate Densities of Elusive Animals,” *Conservation Biology*, 25, 356–364.
- Kéry, M., Royle, J., and Schmid, H. (2005), “Modeling avian abundance from replicated counts using binomial mixture models,” *Ecological Applications*, 15, 1450–1461.
- Kery, M. and Schaub, M. (2011), *Bayesian Population Analysis Using WinBugs*, Academic Press.
- King, R. (2009), “Missing,” *missing*, Missing.
- (missing), “Missing,” *missing*, Missing.
- Kumar (missing), “Unpublished data,” .
- Kuo, L. and Mallick, B. (1998), “Variable selection for regression models,”

- 6337 *Sankhyā*: The Indian Journal of Statistics, Series B, 65–81.
- 6338 Laird, N. M. and Ware, J. H. (1982), “Random-effects models for longitudinal
6339 data,” *Biometrics*, 963–974.
- 6340 Langtimm (2010), “Missing,” *Missing*, missing.
- 6341 Le Cam, L. (1990), “Maximum likelihood: an introduction,” *International Statis-
6342 tical Review/Revue Internationale de Statistique*, 153–171.
- 6343 Lewin-Koh, N. J., Bivand, R., contributions by Edzer J. Pebesma, Archer, E.,
6344 Baddeley, A., Bibiko, H.-J., Dray, S., Forrest, D., Friendly, M., Giraudoux, P.,
6345 Golicher, D., Rubio, V. G., Hausmann, P., Hufthammer, K. O., Jagger, T.,
6346 Luque, S. P., MacQueen, D., Niccolai, A., Short, T., Stabler, B., and Turner,
6347 R. (2011), *maptools: Tools for reading and handling spatial objects*, r package
6348 version 0.8-10.
- 6349 Link, W. A. (2003), “Missing,” *missing*, missing.
- 6350 Link, W. A. and Barker, R. J. (2009), *Bayesian Inference: With Ecological Appli-
6351 cations*, London, UK: Academic Press.
- 6352 Liu and Wu (1999), “Parameter expansion for data augmentation,” *J Am Stat
6353 Assoc*, 94, 1264–1274.
- 6354 MacEachern, S. and Berliner, L. (1994), “Subsampling the Gibbs sampler,” *Amer-
6355 ican Statistician*, 188–190.
- 6356 MacKenzie, D. I., Nichols, J. D., Lachman, G. B., Droege, S., Royle, J. A., and
6357 Langtimm, C. A. (2002), “Estimating site occupancy rates when detection prob-
6358 abilities are less than one,” *Ecology*, 83, 2248–2255.
- 6359 Mackenzie, D. I. and Royle, J. (2005), “Designing occupancy studies: general advice
6360 and allocating survey effort,” *Journal of Applied Ecology*, 42, 1105–1114.
- 6361 Magoun, A. J., Long, C. D., Schwartz, M. K., Pilgrim, K. L., Lowell, R. E., and
6362 Valkenburg, P. (2011), “Integrating motion-detection cameras and hair snags for
6363 wolverine identification,” *The Journal of Wildlife Management*, 75, 731–739.
- 6364 McCarthy, M. A. (2007), *Bayesian Methods for Ecology*, Cambridge: Cambridge
6365 University Press.
- 6366 McCullagh, P. and Nelder, J. (1989), *Generalized linear models*, Chapman &
6367 Hall/CRC.
- 6368 Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., et al. (1953),
6369 “Equation of state calculations by fast computing machines,” *The journal of
6370 chemical physics*, 21, 1087–1092.
- 6371 Metropolis, N. and Ulam, S. (1949), “The Monte Carlo method,” *Journal of the
6372 American Statistical Association*, 44, 335–341.
- 6373 Millar, R. (2009), “Comparison of hierarchical Bayesian models for overdispersed
6374 count data using DIC and Bayes’ Factors,” *Biometrics*, 65, 962–969.
- 6375 Mollet, P., Kery, M., Gardner, B., Pasinelli, G., and A, R. J. (2012), “Population
6376 size estimation for capercaillie (*Tetrao urogallus* L.) using DNA-based individual
6377 recognition and spatial capture-recapture models,” *missing*, missing, missing.
- 6378 Neal, R. (2003), “Slice sampling,” *Annals of Statistics*, 31, 705–741.
- 6379 Nelder, J. and Wedderburn, R. (1972), “Generalized linear models,” *Journal of the*

- 6380 *Royal Statistical Society. Series A (General)*, 370–384.
- 6381 Norris III, J. L. and Pollock, K. H. (1996), “Nonparametric MLE under two closed
6382 capture-recapture models with heterogeneity,” *Biometrics*, 639–649.
- 6383 Pebesma, E. and Bivand, R. (2011), *Package ‘sp’*, r package version 0.9-91.
- 6384 Pledger, S. (2000), “Unified maximum likelihood estimates for closed capture-
6385 recapture models using mixtures,” *Biometrics*, 434–442.
- 6386 Plummer, M. (2003), “JAGS: A program for analysis of Bayesian graphical models
6387 using Gibbs sampling,” in *Proceedings of the 3rd International Workshop on
6388 Distributed Statistical Computing (DSC 2003)*. March, pp. 20–22.
- 6389 — (2009), “rjags: Bayesian graphical models using mcmc. R package version 1.0.
6390 3-12,” .
- 6391 — (2011), *rjags: Bayesian graphical models using MCMC*, r package version 3-5.
- 6392 Plummer, M., Best, N., Cowles, K., and Vines, K. (2006), “CODA: Convergence
6393 Diagnosis and Output Analysis for MCMC,” *R News*, 6, 7–11.
- 6394 Robert, C. P. and Casella, G. (2004), *Monte Carlo statistical methods*, New York,
6395 USA: Springer.
- 6396 — (2010), *Introducing Monte Carlo Methods with R*, New York, USA: Springer.
- 6397 Roberts, G. O. and Rosenthal, J. S. (1998), “Optimal scaling of discrete approxi-
6398 mations to Langevin diffusions,” *Journal of the Royal Statistical Society: Series
6399 B (Statistical Methodology)*, 60, 255–268.
- 6400 Royle, J. (2006), “Site occupancy models with heterogeneous detection probabili-
6401 ties,” *Biometrics*, 62, 97–102.
- 6402 — (2009), “Analysis of capture-recapture models with individual covariates using
6403 data augmentation,” *Biometrics*, 65, 267–274.
- 6404 Royle, J. and Dorazio, R. (2006), “Hierarchical models of animal abundance and
6405 occurrence,” *Journal of Agricultural, Biological, and Environmental Statistics*,
6406 11, 249–263.
- 6407 — (2008), *Hierarchical modeling and inference in ecology: the analysis of data from
6408 populations, metapopulations and communities*, Academic Press.
- 6409 — (2010), “Missing,” *Missing*, missing, missing.
- 6410 — (2011), “Missing,” *Missing*, missing, missing.
- 6411 Royle, J., Dorazio, R., and Link, W. (2007), “Analysis of multinomial models
6412 with unknown index using data augmentation,” *Journal of Computational and
6413 Graphical Statistics*, 16, 67–85.
- 6414 Royle, J. and Dubovsky, J. (2001), “Modeling spatial variation in waterfowl band-
6415 recovery data,” *The Journal of wildlife management*, 726–737.
- 6416 Royle, J., Karanth, K., Gopalaswamy, A., and Kumar, N. (2009), “Bayesian infer-
6417 ence in camera trapping studies for a class of spatial capture-recapture models,”
6418 *Ecology*, 90, 3233–3244.
- 6419 Royle, J. and Link, W. (2006), “Generalized site occupancy models allowing for
6420 false positive and false negative errors,” *Ecology*, 87, 835–841.
- 6421 Royle, J. and Nichols, J. (2003), “Estimating abundance from repeated presence-
6422 absence data or point counts,” *Ecology*, 84, 777–790.

- Royle, J. A. (2004a), “Generalized estimators of avian abundance from count survey data,” *Animal Biodiversity and Conservation*, 27, 375–386.
- (2004b), “Missing,” *Missing*, missing.
- (2008), “Modeling individual effects in the Cormack–Jolly–Seber model: a state-space formulation,” *Biometrics*, 64, 364–370.
- (2010), “missing,” *missing*, missing.
- Royle, J. A., Converse, S., and Link, W. (2011a), “Data augmentation for structured populations,” *unpublished*.
- Royle, J. A., Kéry, M., and Guélat, J. (2011b), “Spatial capture-recapture models for search-encounter data,” *Methods in Ecology and Evolution*, 1–10.
- Royle, J. A., Magoun, A. J., Gardner, B., Valkenburg, P., and Lowell, R. E. (2011c), “Density estimation in a wolverine population using spatial capture-recapture models,” *The Journal of Wildlife Management*, 75, 604–611.
- Royle, J. A. and Young, K. V. (2008), “A Hierarchical Model For Spatial Capture-Recapture Data,” *Ecology*, 89, 2281–2289.
- Sanathanan, L. (1972), “Estimating the size of a multinomial population,” *The Annals of Mathematical Statistics*, 142–152.
- Schofield and Barker (missing), “Missing,” *Missing*, missing.
- Sepúlveda, M., Bartheld, J., Monsalve, R., Gómez, V., and Medina-Vogel, G. (2007), “Habitat use and spatial behaviour of the endangered Southern river otter (*Lontra provocax*) in riparian habitats of Chile: conservation implications,” *Biological Conservation*, 140, 329–338.
- Sillett (2011), “Missing,” *Missing*, missing.
- Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2003), *WinBUGS User Manual Version 1.4*.
- Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and Van Der Linde, A. (2002), “Bayesian measures of model complexity and fit,” *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 583–639.
- Stabler, B. (2006), *shapefiles: Read and Write ESRI Shapefiles*, r package version 0.6.
- Sturtz, S., Ligges, U., and Gelman, A. (2005), “R2WinBUGS: A Package for Running WinBUGS from R,” *Journal of Statistical Software*, 12, 1–16.
- Su, Y.-S. and Yajima, M. (2011), *R2jags: A Package for Running jags from R*, r package version 0.02-17.
- Tanner, M. A. and Wong, W. H. (1987), “The calculation of posterior distributions by data augmentation,” *J Am Stat Assoc*, 82, 528–540.
- Thomas, A., O’Hara, B., Ligges, U., and Sturtz, S. (2006), “Making BUGS Open,” *R News*, 6, 12–17.
- Trolle, M. and Kéry, M. (2005), “Camera-trap study of ocelot and other secretive mammals in the northern Pantanal,” *Mammalia*, 69, 409–416.
- Tyre, A. J., Tenhumberg, B., Field, S. A., Niejalke, D., Parris, K., and Possingham, H. P. (2003), “Improving precision and reducing bias in biological surveys: estimating false-negative error rates,” *Ecological Applications*, 13, 1790–1801.

-
- 6466 Wegan (missing), “missing,” *missing*, missing.
6467 Yang, H. C. and Chao, A. (2005), “Modeling Animals’ Behavioral Response by
6468 Markov Chain Models for Capture–Recapture Experiments,” *Biometrics*, 61,
6469 1010–1017.
6470 Zuur, A., Ieno, E., Walker, N., Saveliev, A., and Smith, G. (2009), *Mixed effects*
6471 *models and extensions in ecology with R*, Springer Verlag.