Chapter 1

Introduction

Chapter 2

GLMS and WinBUGS

Chapter 3

Closed population models

Chapter 4

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.

4.1 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" henceforth[1] so that we may use that model as a point of reference without having to provide a long-winded enumeration of

[1]RC: 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 showin g example data (and latent variables) in a table too

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.

4.2 The binomial observation model

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

$$y_{ij} \sim \text{Bin}(K, p_{ij}) \tag{4.1}$$

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 nind x nrep 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 individuals 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:

$$\text{logit}(p_{ij}) = \alpha_0 + \theta * ||s[i] - x[j]|| \tag{4.2}$$

where, here, $||\mathbf{s}_i - \mathbf{x}_j||$ is the distance between $\mathbf{s}_i$ and $\mathbf{x}_j$. We sometimes write $||\mathbf{s}_i - \mathbf{x}_j|| = dist(\mathbf{s}_i, \mathbf{x}_j) = d_{ij}$. 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

$$\log(p_{ij}) = \log(p_0) - \theta * ||\mathbf{s}_i - \mathbf{x}_j||^2 \tag{4.3}$$

Whatever model encounter probability we choose, we should always keep in mind that the model is described conditional on $\mathbf{s}_i$, which is an unobserved random

variable. Thus, to be precise about this, we should write the observation model as

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

Note that we probably expect that the parameter θ in Eq. 4.2 or 4.3 should be negative, so that the probability of encounter decreases with distance between the trap and individual home range center. The joint likelihood for the data, conditional on the collection of individual activity centers, can therefore be expressed as

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

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

4.2.1 Distance as a latent variable

If we knew precisely every $\mathbf{s}_i$ in the population (and how many, N), then the model specified by eqs. 4.1 and 4.2 or 4.3 is just an ordinary logistic regression type of a model which we learned how to fit using **WinBUGS** previously (chapt. 2), with a covariate d_{ij}. However, the activity centers are unobservable even in the best possible circumstances. In that case, d_{ij} is an unobserved variable, analogous to classical "random effects" models. We need to therefore extend the model to accommodate these random variables with an additional model component. A standard, and perhaps not unreasonable, assumption is the so-called "uniformity assumption" which is to say that the $\mathbf{s}_i$ are uniformly distributed over space (the obvious next question "which space?" is addressed below). This uniformity assumption amounts to a uniform prior distribution on $\mathbf{s}_i$, i.e., the pdf of $\mathbf{s}_i$ is constant, which we may express

$$\Pr(\mathbf{s}_i) proptoconst \tag{4.4}$$

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

(1) Observation model: $y_{ij}|\mathbf{s}_i \sim \text{Bin}(K, p_{ij})$

(2) Encounter probability: $\text{logit}(p_{ij}) = \alpha_0 + \theta * ||\mathbf{s}_i - \mathbf{x}_j||$

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

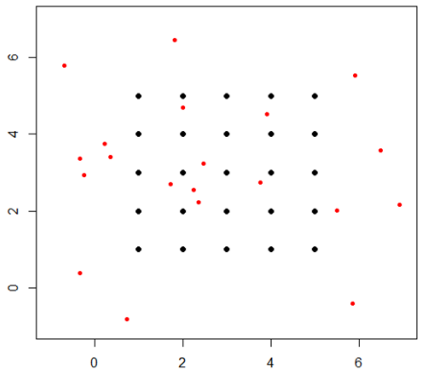

Figure 4.1: Realization of a binomial point process

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

4.3 The Binomial Point-process Model

The collection of individual activity centers $\mathbf{s}_1, \ldots, \mathbf{s}_N$ represent a realization of a *binomial point process* (Illian, 2008, p. xyz). The binomial point process (BPP) is analogous to a Poisson point process in the sense that it represents a "random scatter" of points in space - except that the total number of points is *fixed*, whereas, in a Poisson point process it is random (having a Poisson distribution). As an example, we show in Fig. 4.1 locations of 20 individual activity centers (black dots) in relation to a grid of 25 traps. For a Poisson point process the number of such points in the prescribed state-space would be random whereas often we will simulate fixed numbers of points, e.g., for evaluating the performance of procedures 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 $n(A_1), \ldots, n(A_k)$ in any set of disjoint regions say $A_1, \ldots, A_k$, then these counts are *not* independent. In fact, they have a multinomial distribution (see Illian, 2008, 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. 4.1 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.

4.3.1 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.

4.3.2 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 $\mathbf{s}_i, \ldots, \mathbf{s}_N$ 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-vident 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 3, 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 4.7 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 4.9) 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).

4.3.3 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., $E[n(lastcell)] = 0$. [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 4.7 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 5) 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;

4.3.4 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,

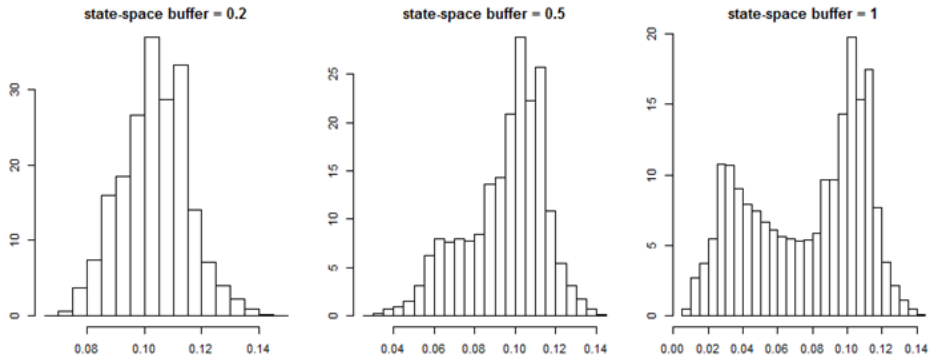

Figure 4.2: Needs a caption

we have a random effect with some prior distribution:

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

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. 4.2 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 $(g, \mathcal{S})$.

4.3.5 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.

4.4 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))
Dmat<-e2dist(traplocs,traplocs) # in cases where speed doesn't matter, it might be
                                # clearer to just show the slow for-loop.
                                # Plus, people will want to copy/paste this stuff
ntraps<-nrow(traplocs)

# define state-space of point process. (i.e., where animals live).
# "delta" just adds a fixed buffer to the outer extent of the traps.
delta<-2
Xl<-min(traplocs[,1] - delta)
Xu<-max(traplocs[,1] + delta)
Yl<-min(traplocs[,2] - delta)
```

```
Yu<-max(traplocs[,2] + delta)

N<-100   # population size
K<- 20    # number nights of effort

sx<-runif(N,Xl,Xu)    # simulate activity centers
sy<-runif(N,Yl,Yu)
S<-cbind(sx,sy)
D<- e2dist(S,traplocs)  # distance of each individual from each trap

alpha0<- -2.5      # define parameters of encounter probability
sigma<- 0.5        #
theta<- 1/(2*sigma*sigma)
probcap<- expit(-2.5)*exp( - theta*D*D)    # probability of encounter
# now generate the encounters of every individual in every trap
Y<-matrix(NA,nrow=N,ncol=ntraps)
for(i in 1:nrow(Y)){
   Y[i,]<-rbinom(ntraps,K,probcap[i,])
}
```

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

```
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.

```
> names(data)
[1] "Y"         "traplocs" "xlim"     "ylim"     "N"         "alpha0"   "beta"
[8] "sigma"    "K"
> Y<-data$Y
> traplocs<-data$traplocs
```

4.4.1 Formatting and manipulating real data sets

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

(1) The basic 2-dimensional data format, which is an `nind` × `ntraps` encounter frequency matrix such as that simulated previously;

(2) The maximally informative 3-dimensional array which we establish here the convention that it has dimensions `nind` × `nperiods` × `ntraps` and

(3) We use a compact format - the "SCR flat format" - which we describe below in section 4.7.

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

```
Y<-array(NA,dim=c(N,K,ntraps))
for(i in 1:nrow(Y)){
for(j in 1:ntraps){
 Y[i,1:K,j]<-rbinom(K,1,probcap[i,j])
}
}
```

We see that a collection of K binary encounter events are generated for *each* individual and for *each* trap. The probabilities have those Bernoulli trials are computed based on the distance from each individuals home range center and the trap (see calculation above), and those are housed in the matrix probcap. Our data simulator function `simSCR0.fn` will return the full 3-d array if `array3d=TRUE` is specified in the function call. To recover the 2-d matrix from the 3-d array, and subset the 3-d array to individuals that were captured, we do this:

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

4.5 Fitting an SCR Model in BUGS

Clearly if we somehow knew the value of N then we could fit this model directly because, in that case, it is a special kind of logistic regression model - one with a random effect, but that enters into the model in a peculiar fashion - and also with 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.

```
data<-simSCR0.fn(discard0=FALSE,sd=2013)
y<-data$Y
traplocs<-data$traplocs
nind<-nrow(y)
X<-data$traplocs
```

```
J<-nrow(X)
y<-rbind(y,matrix(0,nrow=(100-nrow(y)),ncol=J ) )
Xl<-data$xlim[1]
Yl<-data$ylim[1]
Xu<-data$xlim[2]
Yu<-data$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.

```
cat("
model {
alpha0~dnorm(0,.1)
logit(p0)<- alpha0
theta~dnorm(0,.1)
for(i in 1:N){
 s[i,1]~dunif(Xl,Xu)
 s[i,2]~dunif(Yl,Yu)
for(j in 1:J){
d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
y[i,j] ~ dbin(p[i,j],K)
p[i,j]<- p0*exp(- theta*d[i,j]*d[i,j])
}
}

}
",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 $\mathbf{s}$ 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

```
sst<-cbind(runif(nind,Xl,Xu),runif(nind,Yl,Yu))  # starting values for s
for(i in 1:nind){
if(sum(y[i,])==0) next
sst[i,1]<- mean( X[y[i,]>0,1] )
sst[i,2]<- mean( X[y[i,]>0,2] )
}

data <- list (y=y,X=X,K=K,N=nind,J=J,Xl=Xl,Yl=Yl,Xu=Xu,Yu=Yu)
```

```
inits <- function(){
  list (alpha0=rnorm(1,-4,.4),theta=runif(1,1,2),s=sst)
}

library("R2WinBUGS")
parameters <- c("alpha0","theta")
nthin<-1
nc<-3
nb<-1000
ni<-2000
out <- bugs (data, inits, parameters, "SCR0a.txt", n.thin=nthin,
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:

```
> print(out,digits=3)
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    -2.496  0.224  -2.954  -2.648  -2.48  -2.340  -2.091 1.013   190
theta      2.442  0.419   1.638   2.145   2.44   2.721   3.303 1.005   530
deviance 292.803 21.155 255.597 277.500 291.90 306.000 339.302 1.006   380

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

DIC info (using the rule, pD = Dbar-Dhat)
pD = -138.8 and DIC = 154.0
DIC is an estimate of expected predictive error (lower deviance is better).
```

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

4.6 Unknown N

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

We typically have more than 4 traps and, if we're fortunate, many more individuals in our data set.

For the augmented data, we introduce a set of binary latent variables (the data augmentation variables), z_i, and the model is extended to describe $\Pr(z_i = 1)$ 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. 6 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 $D = N/64$.

4.6.1 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
theta~dnorm(0,.1)
psi~dunif(0,1)

for(i in 1:M){
 z[i] ~ dbern(psi)
 s[i,1]~dunif(Xl,Xu)
 s[i,2]~dunif(Yl,Yu)
for(j in 1:J){
d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
y[i,j] ~ dbin(p[i,j],K)
p[i,j]<- z[i]*p0*exp(- theta*d[i,j]*d[i,j])
}
}
N<-sum(z[])
D<-N/64
}
",file = "SCR0a.txt")
```

To prepare our data we have to augment the data matrix `y` with $M - n$

all-zero encounter histories, we have to create starting values for the variables z_i and also the activity centers $\mathbf{s}_i$ of which, for each, we require M values. Otherwise the remainder of the code for bundling the data, creating initial values and executing **WinBUGS** looks much the same as before except with more or differently named arguments.

```
## Data augmentation stuff
M<-200
y<-rbind(y,matrix(0,nrow=M-nind,ncol=ncol(y)))
z<-c(rep(1,nind),rep(0,M-nind))

sst<-cbind(runif(M,Xl,Xu),runif(M,Yl,Yu))  # starting values for s
for(i in 1:nind){
if(sum(y[i,])==0) next
sst[i,1]<- mean( X[y[i,]>0,1] )
sst[i,2]<- mean( X[y[i,]>0,2] )
}
data <- list (y=y,X=X,K=K,M=M,J=J,Xl=Xl,Yl=Yl,Xu=Xu,Yu=Yu)
inits <- function(){
  list (alpha0=rnorm(1,-4,.4),theta=runif(1,1,2),s=sst,z=z)
}

library("R2WinBUGS")
parameters <- c("alpha0","theta","N")
nthin<-1
nc<-3
nb<-1000
ni<-2000
out <- bugs (data, inits, parameters, "SCR0a.txt", n.thin=nthin,n.chains=nc,
 n.burnin=nb,n.iter=ni,debug=TRUE,working.dir=getwd())
```

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

```
> print(out1,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    -2.57  0.23  -3.04  -2.72  -2.56  -2.41  -2.15 1.01   320
theta      2.46  0.42   1.63   2.16   2.46   2.73   3.33 1.02   120
N        113.62 15.73  86.00 102.00 113.00 124.00 147.00 1.01   260
D          1.78  0.25   1.34   1.59   1.77   1.94   2.30 1.01   260
deviance 302.60 23.67 261.19 285.47 301.50 317.90 354.91 1.00  1400
```

```
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 = 279.9 and DIC = 582.5
DIC is an estimate of expected predictive error (lower deviance is better).
```

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

```
> 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).
```

4.6.2 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`.

4.7 Case Study: Wolverine Camera Trapping Study

We provide an analysis here of A. Magoun's wolverine data (Magoun et al., 2011; Royle et al., 2011). The study took place in SE Alaska (Fig. 4.3) 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. 4.1) 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. 4.6 accordingly. Our treatment of the data here is based on the analysis of Royle et al. (2011).

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. 4.4.1 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
 [1,]      1          2    127
 [2,]      1          2    128
 [3,]      1          2    129
 [4,]      1         18    130
 [5,]      2          3    106
```

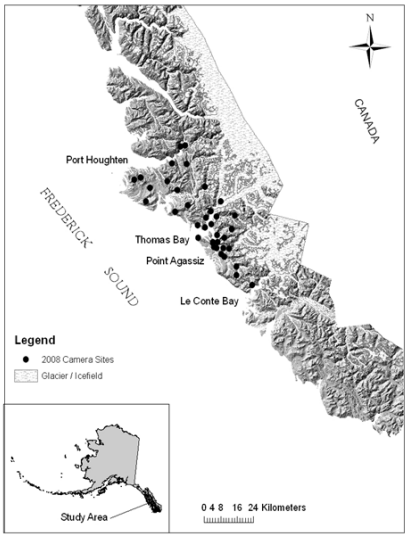

Figure 4.3: Wolverine camera trap locations from Magoun et al. (2011).

```
   [6,]      2          18     104
   [7,]      5           5      73
   [8,]      5           5      89
   [9,]      6          18     117
  [10,]      6          18     118
```

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

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

The "encounter data file" `wcaps.csv` exists in the **R** package `scrbook` as a .csv file that people can read into **R** and do some basic summary statistics on.

For our purposes we need to convert these data into the "individual x trap" array of binary encounter frequencies, although more general models might require an encounter-history formulation of the model which requires a full 3-d array. To obtain our nind x ntrap encounter frequency matrix, we do this the hard way by first converting the encounter data file into a 3-d array and then summarize to trap totals. We have a handy function `SCR23darray.fn` which takes the compact encounter data file with optional arguments ntraps and nperiods, and converts it to a 3-d array, and then we use the **R** function `apply` to summarize over the "sample" period dimension (by convention here, this is the 2nd dimension):

```
SCR23darray.fn <- function(caps,ntraps=NULL,nperiods=NULL){
nind<-max(caps[,2])
if(is.null(ntraps)) ntraps<-max(caps[,1])
if(is.null(nperiods)) nperiods<- max(caps[,3])

y<-array(0,c(nind,nperiods,ntraps))
tmp<-cbind(caps[,2],caps[,3],caps[,1])
y[tmp]<-1
y
}

# for the wolverine data do this:

Y3d <-SCR23darray.fn(wcaps,ntraps=37,nperiods=165)
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:

```
      Trap Easting Northing 1 2 3 4 5 6 7 <- column names
         1   39040    19216 0 0 0 0 0 0 0
         2   41324    19772 1 1 1 1 1 1 1
         3   44957    12985 0 0 0 0 0 0 0
         4   41151    23220 0 0 0 0 0 0 0
         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:

```
traps<- read.csv("wtraps.csv")
traplocs<- traps[,2:3]
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. (2011) which shows the trap and total encounter frequencies:

Table 4.1: Individual frequencies of capture for wolverines captured in camera traps in Southeast 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

4.7.1 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, $\alpha_0 = logit(p_0)$, θ and ψ. It remains to define the state-space $\mathcal{S}$. For this, we nested the trap array (Fig. 4.3) in a 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 ($area = 10374km * km$) 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):

```
Buffer = 10 km
> print(out1$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
           mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi        0.11  0.02   0.07   0.10   0.11   0.13   0.17    1  2400
sigma      1.79  0.29   1.31   1.58   1.75   1.97   2.46    1   600
p0         0.03  0.00   0.02   0.03   0.03   0.03   0.04    1 13000
N         33.02  4.99  25.00  29.00  32.00  36.00  44.00    1  1600
D          4.93  0.75   3.73   4.33   4.78   5.38   6.57    1  1600
beta       0.17  0.05   0.08   0.13   0.16   0.20   0.29    1   600
deviance 441.97 11.49 421.50 434.00 441.20 449.20 466.30    1  6600


Buffer = 20 km
> print(out2$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
```

```
            mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi         0.16  0.04   0.10   0.13   0.16   0.18   0.24    1  4200
sigma       1.78  0.32   1.29   1.55   1.73   1.94   2.56    1 20000
p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04    1  3000
N          47.40  9.19  32.00  41.00  46.00  53.00  68.00    1  5900
D           4.57  0.89   3.08   3.95   4.43   5.11   6.55    1  5900
beta        0.17  0.06   0.08   0.13   0.17   0.21   0.30    1 20000
deviance  444.36 11.84 423.60 436.00 443.60 451.80 469.70    1  1800

Buffer = 25 km
> print(out3$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
            mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi         0.19  0.04   0.11   0.16   0.19   0.22   0.29 1.00   790
sigma       1.80  0.34   1.30   1.56   1.75   1.98   2.59 1.01   400
p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04 1.00  2800
N          56.66 11.47  37.00  48.00  56.00  64.00  82.00 1.00   570
D           4.53  0.92   2.96   3.84   4.48   5.11   6.55 1.00   570
beta        0.17  0.06   0.07   0.13   0.16   0.20   0.30 1.01   400
deviance  444.75 11.87 423.60 436.40 444.00 452.30 469.80 1.00 24000

Buffer = 30 km
> print(out4$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
            mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi         0.23  0.05   0.14   0.19   0.22   0.26   0.34 1.00  1500
sigma       1.79  0.34   1.29   1.55   1.73   1.97   2.58 1.01   560
p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04 1.00 30000
N          67.39 14.12  43.00  57.00  66.00  76.00  98.00 1.00  1200
D           4.54  0.95   2.90   3.84   4.44   5.12   6.60 1.00  1200
beta        0.17  0.06   0.07   0.13   0.17   0.21   0.30 1.01   560
deviance  444.58 11.83 423.60 436.40 443.80 452.20 469.90 1.00  4700

Buffer = 40 km (need to add this)



Buffer = 45 km
> print(out7$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
            mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi         0.36  0.08   0.21   0.30   0.35   0.41   0.53    1  5000
sigma       1.78  0.34   1.29   1.55   1.72   1.95   2.60    1   850
p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04    1  3600
```

```
N         106.57 23.34  67.00  90.00 104.00 121.00 157.00    1  3400
D           4.62  1.01   2.90   3.90   4.51   5.25   6.81    1  3400
beta        0.17  0.06   0.07   0.13   0.17   0.21   0.30    1   850
deviance  444.80 11.84 423.60 436.40 444.10 452.30 470.00    1 30000

Buffer = 50 km
> print(out8$out,digits=2)
Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
 3 chains, each with 12000 iterations (first 2000 discarded)
 n.sims = 30000 iterations saved
             mean    sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
psi         0.40  0.09   0.23   0.33   0.39   0.45   0.60 1.01  1300
sigma       1.82  0.48   1.30   1.56   1.74   1.97   2.68 1.05   200
p0          0.03  0.00   0.02   0.03   0.03   0.03   0.04 1.00  5800
N         118.47 26.81  71.00 100.00 117.00 135.00 176.00 1.01  1200
D           4.52  1.02   2.71   3.82   4.46   5.15   6.72 1.01  1200
beta        0.17  0.06   0.07   0.13   0.17   0.21   0.30 1.05   200
deviance  444.84 11.90 423.90 436.50 444.10 452.20 470.30 1.00   500
```

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

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

4.7.2 Conclusion of Analysis

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

It is worth thinking about this model, and these estimates, computed under a rectangular state space roughly centered over the trapping array (Fig. 4.3). Does it make sense to define the state-space to include, for example, ocean? What are the possible consequences of this? What can we do about it? There's no reason at all that the state space has to be a regular polygon – we defined it as such here strictly for convenience and for ease of implementation in **WinBUGS**

where it enables us to specify the prior for the activity centers as uniform priors for each coordinate. While it would be possible to define a more realistic state-space using some general polygon, it might take some effort to implement that in the **BUGS** language (see chapter XYZXYZ[2] for example of a simple case). Alternatively, we recommend using a discrete representation of the state-space – i.e., approximate $\mathcal{S}$ by a grid of G points. We discuss this in the following section.

4.8 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 $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$. 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 $m = 1, 2, \ldots,$ `niter` and compute this summary:

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

Thus, $N(x, 1), N(x, 2), \ldots,$ 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:

[2] raccoon example or something?

```
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 $N \times$ `niter` matrices:

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

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

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

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

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

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

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

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

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

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

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

```
spatial.plot<- function(x,y){
 nc<-as.numeric(cut(y,20))
 plot(x,pch=" ")
 points(x,pch=20,col=topo.colors(20)[nc],cex=2)
 image.scale(y,col=topo.colors(20))
}
# To execute the function do this:
spatial.plot(cbind(xg,yg), Dn/nrow(z))
```

4.8.1 Example: Wolverine density map.

We used the posterior output from the wolverine model fitted previous to compute a relatively coarse version of a density map, using a 10×10 grid (Fig. 4.4) and using a 30×30 grid (Fig. 4.5)[3]. In these figures density is expressed in units of individuals per 1000 km^2, while the area of the pixels is about 1037 km^2 and 115 km^2, respectively. That calculation is based on[4]:

```
> total.area<- (Yu-Yl)*(Xu-Xl)*1000
> total.area/(10*10)
[1] 1037.427
> total.area/(30*30)
[1] 115.2697
```

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

4.9 Discrete State-Space

The SCR model developed previously in this chapter assumes that individual activity centers are distributed uniformly over the prescribed state-space. Clearly

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

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

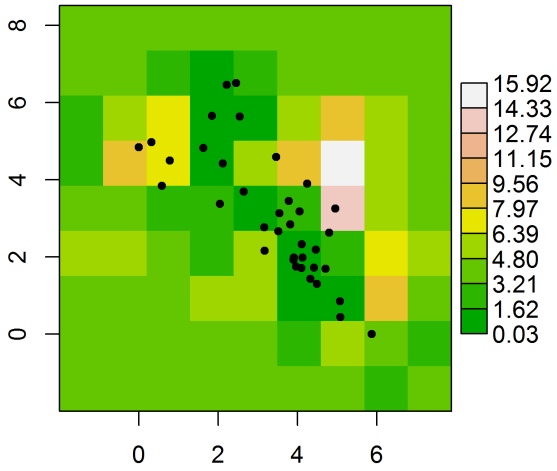

Figure 4.4: Needs a caption

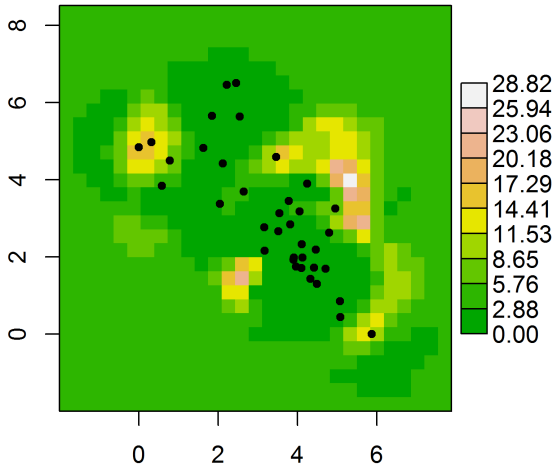

Figure 4.5: Needs a caption

this will not always be a reasonable assumption. In chapter 7 we talk about developing 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 $\mathbf{s}_1, \ldots, \mathbf{s}_G$, 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 6 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 6. 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 5, 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 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).

4.9.1 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 4.3.4 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 5) 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 XYZ[5].

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

Note: WinBUGS based on fewer samples too!

To get SE and time-series SE do this:

[5] Andy to finish later

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

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

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

4.9.2 Analysis of the wolverine camera trapping data

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

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

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

2km
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).
       mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
psi    0.28  0.06  0.17  0.24  0.27  0.32   0.41 1.01   230
sigma  0.64  0.05  0.55  0.60  0.64  0.67   0.73 1.02    88
lam0  -3.00  0.16 -3.33 -3.11 -3.00 -2.90  -2.69 1.04    52
p0     0.05  0.01  0.03  0.04  0.05  0.05   0.06 1.04    52
N     82.95 16.26 55.00 72.00 82.00 93.00 119.02 1.01   240

4 km
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).
       mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
psi    0.30  0.06  0.19  0.26  0.29  0.34   0.43 1.01   580
```

```
sigma  0.62  0.05  0.54  0.59  0.62  0.65   0.72 1.00  2000
lam0  -3.00  0.16 -3.33 -3.10 -2.99 -2.90  -2.67 1.01   390
p0     0.05  0.01  0.03  0.04  0.05  0.05   0.06 1.01   390
N     88.78 16.76 60.00 77.00 87.00 99.00 125.00 1.01   690

8km
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).
       mean    sd  2.5%   25%   50%   75%  97.5% Rhat n.eff
psi    0.27  0.06  0.17  0.23  0.27  0.31   0.40 1.00  1500
sigma  0.69  0.05  0.60  0.65  0.68  0.72   0.80 1.00  3000
lam0  -3.07  0.17 -3.41 -3.20 -3.07 -2.95  -2.74 1.01   210
p0     0.04  0.01  0.03  0.04  0.04  0.05   0.06 1.01   200
N     82.01 15.98 55.00 71.00 80.00 92.00 118.00 1.00  1300
```

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

```
2km
Iterations = 7001:13000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 6000

          Mean        SD  Naive SE Time-series SE
N     86.28522 16.950626 1.263e-01      0.4878973
lam0   0.04807  0.007512 5.599e-05      0.0002199
p0     0.04581  0.006820 5.083e-05      0.0001996
psi    0.28904  0.062117 4.630e-04      0.0017481
sigma  0.62769  0.043596 3.249e-04      0.0018724

4km
          Mean        SD  Naive SE Time-series SE
N     85.53139 16.998966 1.267e-01      0.5181297
lam0   0.04636  0.007542 5.621e-05      0.0002382
p0     0.04425  0.006867 5.118e-05      0.0002172
psi    0.28650  0.061922 4.615e-04      0.0018276
sigma  0.64281  0.048321 3.602e-04      0.0022911

8km
          Mean        SD  Naive SE Time-series SE
N     83.97039 16.508146 1.230e-01      0.4548782
lam0   0.04519  0.006919 5.157e-05      0.0001738
p0     0.04319  0.006319 4.710e-05      0.0001589
psi    0.28146  0.060653 4.521e-04      0.0016555
sigma  0.66956  0.040989 3.055e-04      0.0015070
```

4.9.3 SCR models as multi-state models

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

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

4.10 Summary and Outlook

A point we tried to emphasize in this chapter is that the basic SCR model is not much more than an ordinary capture-recapture model for closed populations – it is simply that model but augmented with a set of "individual effects", $\mathbf{s}_i$, which relate encounter probability to some sense of individual location. SCR models are therefore a type of individual covariate model (as introduced in chapter 3 – but with imperfect information about the individual covariate. In other words, they are GLMM type models when N is known or, when N is unknown, they are zero-inflated GLMMs (see Royle (2006)). Another class of capture-recapture models that SCR models are closely related to is so-called "Model M_h." The effect of 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 5.

Formal consideration of the collection of individual locations $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$ 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 stat-espace is part of the model. It can have an influence on parameter estimates and other inferences such as model selection (see chapter **??**). 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 including 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 **??**. 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 **??**), spatial covariates on density (chapter 7), open populations (chapter **??**), model assessment and selection (chapter **??**) and other topics. We also consider technical details of Bayesian (chapter 6) and maximum likelihood (chapter 5) estimation so that the interested reader can develop or extend their own methods to suit their needs.

Chapter 5

Maximum likelihood estimation

Chapter 6

MCMC details

Chapter 7

Inhomogeneous Point Process

Bibliography

Arnason (1973), "Missing," *Missing*, Missing, Missing.

— (1974), "Missing," *Missing*, Missing, Missing.

Borchers, D. and Efford, M. (2008), "Spatially explicit maximum likelihood methods for capture–recapture studies," *Biometrics*, 64, 377–385.

Efford, M. (2004), "Density estimation in live-trapping studies," *Oikos*, 106, 598–610.

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.

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.

Hestbeck (1991), "Missing," *Missing*, Missing, Missing.

Illian (2008), "Missing," *Missing*, Missing.

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.

Kery, M. and Schaub, M. (2011), *Bayesian Population Analysis Using WinBugs*, Academic Press.

Link, W. A. (2003), "Missing," *missing*, missing.

Magoun, A. J., Long, C. D., Schwartz, M. K., Pilgrim, K. L., Lowell, R. E., and Valkenburg, P. (2011), "Integrating motion-detection cameras and hair snags for wolverine identification," *The Journal of Wildlife Management*, 75, 731–739.

Mollet, P., Kery, M., Gardner, B., Pasinelli, G., and A, R. J. (2012), "Population size estimation for capercaille (Tetrao urogallus L.) using DNA-based individual recognition and spatial capture-recapture models," *missing*, missing, missing.

Norris III, J. L. and Pollock, K. H. (1996), "Nonparametric MLE under two closed capture-recapture models with heterogeneity," *Biometrics*, 639–649.

Pledger, S. (2000), "Unified maximum likelihood estimates for closed capture-recapture models using mixtures," *Biometrics*, 434–442.

Plummer, M. (2003), "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling," in *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003). March*, pp. 20–22.

— (2009), "rjags: Bayesian graphical models using mcmc. R package version 1.0. 3-12," .

Royle, J. (2006), "Site occupancy models with heterogeneous detection probabilities," *Biometrics*, 62, 97–102.

Royle, J., Karanth, K., Gopalaswamy, A., and Kumar, N. (2009), "Bayesian inference in camera trapping studies for a class of spatial capture-recapture models," *Ecology*, 90, 3233–3244.

Royle, J. A., Magoun, A. J., Gardner, B., Valkenburg, P., and Lowell, R. E. (2011), "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.

Thomas, A., OHara, B., Ligges, U., and Sturtz, S. (2006), "Making BUGS Open," *R News*, 6, 12–17.