

# Chapter 1

## 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  $s$ , we consider a fully hierarchical model in which we regard  $s$  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.

## 1.1 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  $6 \times 4$  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 individuals can be encountered at most 1 time in a trap during any occasion. This is also typical of 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 (clustering: we discuss this further in Chapter XXX). We suppose also that traps may capture any number of individuals and an individual may be captured in any number of traps during each occasion. Thus, this is the “multi-catch” trap situation (Efford XXXX p. XXXX). 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<sup>1</sup> so that we may use that model as a point of reference without having to provide a long-winded enumeration of assumptions and sampling design each time we do. We will make things more precise as we develop a formal statistical definition of the model shortly.

## 1.2 The binomial observation model

We assume that the individual and trap-specific encounters,  $y[i, j]$ , are mutually independent outcomes of a binomial random variable:

$$y[i, j] \sim \text{Binomial}(K, p[i, j])$$

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  $p[i, j]$  is indexed by (i.e., depends on) both individual and trap. In a sense, then, we can think of each *trap* as producing individual level encounter history data of the classical variety - an  $n \times m$  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  $p[i, j]$  is defined conditional on “where” individual  $i$  lives. Naturally, we think about defining an individual home range and then relating  $p[i, j]$  explicitly to the centroid of the individual's home range, or its center of activity (Efford, 2004; Borchers and Efford, 2008; Royle and Young, 2008). Therefore, define  $s[i]$ , a two-dimensional spatial coordinate, to be the activity center for individual  $i$ . Then, the basic SCR model postulates that encounter probability,  $p[i, j]$ , is related by a decreasing function to distance between trap  $j$ , having location  $x[j]$ , and  $s[i]$ . Naturally, if we think of modeling binomial counts using logistic regression, we might specify the model according to:

$$\text{logit}(p[i, j]) = \alpha + \beta * ||s[i] - x[j]||$$

where, here,  $||s[i] - x[j]||$  is the distance between  $s[i]$  and  $x[j]$ . We sometimes write  $||s[i] - x[j]|| = \text{dist}[s[i], x[j]] = d[i, j]$ . Alternatively, if we think about distance sampling then we might use the “half-normal” model of the form:

$$p[i, j] = p_0 * \exp(-\beta * ||s[i] - x[j]||^2)$$

Or any of a large number of standard detection models that are commonly used. The half-normal model implies

$$\log(p[i, j]) = \log(p_0) - \beta * ||s[i] - x[j]||^2$$

We address other detection models in chapter XYZ and elsewhere.

---

<sup>1</sup>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 showing example data (and latent variables) in a table too

100 Whatever model encounter probability we choose, we should always keep in  
 101 mind that the model is described conditional on  $s[i]$ , which is an unobserved  
 102 random variable. Thus, to be precise about this, we should write the observation  
 103 model as

$$y[i, j] | s[i] \sim \text{Binomial}(K, p(s[i, j]))$$

104 Note that we probably expect that the parameter beta in  $[**]$  should be negative,  
 105 so that the probability of encounter decreases with distance between the trap  
 106 and individual home range center. The joint likelihood for the data, conditional  
 107 on the collection of individual activity centers, can therefore be written as

$$\mathcal{L}(stuff | s[1], \dots, s[N]) = \prod_i \prod_j \text{Bin}(y_{ij} | p_{ij})$$

108 Which, if we switch the indices on the product operators, this shows the SCR  
 109 likelihood (conditional on  $s$ ) to be the product of  $J$  INDEPENDENT capture-  
 110 recapture likelihoods - one for each trap. However, the data have a “repeated  
 111 measures” type of structure, with each of the  $j$  likelihood contributions for each  
 112 individual being “grouped” by individual. Thus, we cannot analyze the model  
 113 meaningfully by  $J$  trap-specific models. In classical repeated measures types of  
 114 models, we accommodate the group structure of the data using random effects  
 115 (random individual or group level variables). For SCR models we take the same  
 116 basic approach, which we develop subsequently.

### 117 1.2.1 Distance as a latent variable

118 If we knew precisely every  $s[i]$  in the population (and how many,  $N$ ), then the  
 119 model specified by  $[*]$  and  $[**]$  is just an ordinary logistic regression type of  
 120 a model which we learned how to fit using WinBUGS previously (Chapter 2),  
 121 with a covariate  $d[i, j]$ . However, the activity centers are unobservable even in  
 122 the best possible circumstances. In that case,  $d[i, j]$  is an unobserved variable,  
 123 analogous to classical “random effects” models. We need to therefore extend the  
 124 model to accommodate these random variables with an additional model com-  
 125 ponent. A standard, and perhaps not unreasonable, assumption is the so-called  
 126 “uniformity assumption” which is to say that the  $s[i]$  are uniformly distributed  
 127 over space (the obvious next question “which space?” is addressed below). This  
 128 uniformity assumption amounts to a uniform prior distribution on  $s[i]$ , i.e., the  
 129 pdf of  $s[i]$  is constant, which we may express

$$\Pr(s[i]) \text{propto} \text{const} \tag{1.1}$$

130 The 3 starred equations completely define the important parts of this spatial  
 131 capture-recapture model. To summarize:

- 132 (1)  $y[i, j] | s[i] \sim \text{Binomial}(K, p[i, j])$
- 133 (2)  $\text{logit}(p[i, j]) = \text{alpha} + \text{beta} * ||s[i] - x[j]||$
- 134 (3)  $\Pr[s[i]] \propto \text{const}$

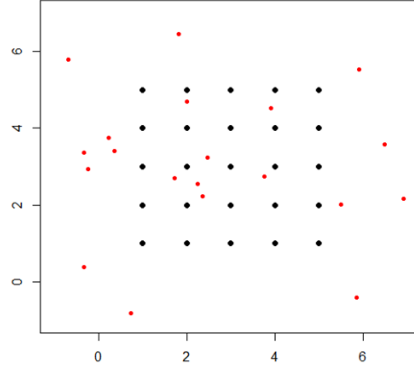


Figure 1.1: Realization of a binomial point process

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”,  $s[i]$ , 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  $s[i]$ . The term “state-space” is a technical way of saying “possible outcomes”.

### 1.3 The Binomial Point-process Model

The collection of individual activity centers  $s[1], \dots, s[N]$  represent a realization of a *binomial point process* (e.g., 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 Figure XYZ 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.

One consequence of having fixed  $N$ , in the BPP model, is that the BPP is not strictly a model of “complete spatial randomness”. This is because if you form counts in any set of regions say  $A_1, \dots, A_k$ , let  $n(A_1), \dots, n(A_k)$  be the counts 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 Figure XYZ (above) 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.

### 1.3.1 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. Furthermore, as we will see in Ch XX, we have some flexibility in how we model an animal’s movement around its activity center.

### 1.3.2 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  $s[1], \dots, s[N]$  are unknown quantities and we will need to be able to simulate each  $s[i]$  in the population from the posterior distribution. It should be self-evident that we cannot simulate the  $s[i]$  unless we describe precisely the region over which those  $s[i]$ '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  $s[i]$ . Thus, an equivalent explicit statement of the “uniformity assumption” is

$$s[i] \text{ Uniform}(\mathcal{S})$$

#### Choosing a 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 xx wolverine xxx 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 that will allow specific points to be deleted or not depending on whether they represent habitat (Chapter XYZ)<sup>2</sup> This is primarily necessary to provide control over the geometry and morphology of what we consider to be suitable habitat (see section XYZ below). In what

<sup>2</sup>Or we can use a polygon with holes, where individuals can't occur. The “point in polygon” approach... perhaps not easy with Gibbs sampling. This should be covered in the MCMC chapter.

follows below we provide an analysis of the camera data defining the state-space to be a regular continuous polygon (a rectangle).

### 1.3.3 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  $S$ , if  $S$  is sufficiently large. In fact, this only holds as long as our model relating  $p[i, j]$  to  $s[i]$  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  $d_i B$  for  $B$  large enough) is 0. So it always contributes nothing to the likelihood, i.e.,  $E[n(\text{lastcell})] = 0$ . [sketch out a proof of this], in regular situations in which the detection function decays monotonically with distance and prior density is constant. Sometimes our estimate of density can be influenced if we make  $S$  too small but this might be sensible if  $S$  is naturally well-defined. As we discussed in chapter 1, **choice of  $S$  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  $S$ .** 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  $S$ , we should default to choosing  $S$  to be very large in order to achieve invariance or otherwise evaluate sensitivity of density estimates by trying a couple of different values of  $S$ . This is a standard “sensitivity to prior” argument that Bayesians always have to be conscious of. We demonstrate this in our analysis of section XXX.YYY below. Note that  $area(S)$  affects data augmentation. If you increase  $area(S)$  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 XXX) we have to integrate the conditional-on-s likelihood over some 2-dimensional space. It might work that the integration can be done from -infinity to +infinity but that is a mathematical artifact of specific detection functions, and an implicit definition of a state-space that doesn’t make biological sense, even though it may in fact be innocuous;



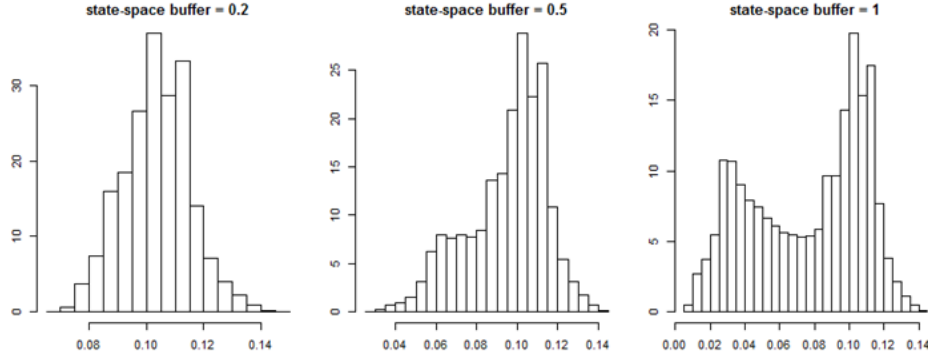


Figure 1.2: Needs a caption

### 1.3.4 Connection to Model Mh

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

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

And  $p(s) = p(y = 1|s)$  is some function of  $s$ . 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  $p(s)$  over the state-space of  $s$ . Obviously the choice of  $p(s)$  and the choice of  $\mathcal{S}$  interact to determine the effective heterogeneity in  $p$ . We show an illustration in Fig. ?? below which shows a histogram of  $p$  for a hypothetical population of 100000 individuals on a state-space enclosing our 5x5 trap array above, under the logistic model for distance. R code is available on the Web Supplement 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.

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

### 311 1.3.5 Connection to Distance Sampling

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

## 326 1.4 Simulating SCR Data

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

```
340 set.seed(2013)
341 # create 5 x 5 grid of trap locations with unit spacing
342 traplocs<- cbind(sort(rep(1:5,5)),rep(1:5,5))
343 Dmat<-e2dist(traplocs,traplocs) # in cases where speed doesn't matter, it might be
344                                # clearer to just show the slow for-loop.
345                                # Plus, people will want to copy/paste this stuff
```

```

346 ntraps<-nrow(traplocs)
347
348 # define state-space of point process. (i.e., where animals live).
349 # "delta" just adds a fixed buffer to the outer extent of the traps.
350 delta<-2
351 Xl<-min(traplocs[,1] - delta)
352 Xu<-max(traplocs[,1] + delta)
353 Yl<-min(traplocs[,2] - delta)
354 Yu<-max(traplocs[,2] + delta)
355
356 N<-100 # population size
357 K<- 20 # number nights of effort
358
359 sx<-runif(N,Xl,Xu) # simulate activity centers
360 sy<-runif(N,Yl,Yu)
361 S<-cbind(sx,sy)
362 D<- e2dist(S,traplocs) # distance of each individual from each trap
363
364 alpha0<- -2.5 # define parameters of encounter probability
365 sigma<- 0.5 #
366 beta<- 1/(2*sigma*sigma)
367 probcap<- expit(-2.5)*exp( - beta*D*D) # probability of encounter
368 # now generate the encounters of every individual in every trap
369 Y<-matrix(NA,nrow=N,ncol=ntraps)
370 for(i in 1:nrow(Y)){
371   Y[i,]<-rbinom(ntraps,K,probcap[i,])
372 }

```

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

```

381 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.

```

384 > names(data)
385 [1] "Y"      "traplocs" "xlim"      "ylim"      "N"      "alpha0"    "beta"
386 [8] "sigma"   "K"
387 > Y<-data$Y
388 > traplocs<-data$traplocs

```

### 389 1.4.1 Formatting and manipulating real data sets

390 Conventional capture-recapture data are easily stored and manipulated as a  
 391 2-dimensional array, an  $n_{ind} \times n_{period}$  matrix, which is maximally informa-  
 392 tive for any conventional capture-recapture model, but not for spatial capture-  
 393 recapture models. For SCR models we must preserve the spatial information in  
 394 the encounter history information. We will routinely analyze data in 3 standard  
 395 formats:

- 396 (1) Basic 2-dimensional data format, which is an  $n_{ind} \times n_{traps}$  encounter  
 397 frequency matrix such as that simulated previously;
- 398 (2) The maximally informative 3-dimensional array which we establish here  
 399 the convention that it has dimensions  $n_{ind} \times n_{periods} \times n_{traps}$  and
- 400 (3) We use a compact format - the “SCR flat format” - which we describe  
 401 below (Section XYZ wolverine XYZ).

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

```
405 Y<-array(NA,dim=c(N,K,ntraps))
406 for(i in 1:nrow(Y)){
407   for(j in 1:ntraps){
408     Y[i,1:K,j]<-rbinom(K,1,probcap[i,j])
409   }
410 }
```

411 We see that a collection of  $K$  binary encounter events are generated for  
 412 *each* individual and for *each* trap. The probabilities have those Bernoulli trials  
 413 are computed based on the distance from each individuals home range center  
 414 and the trap (see calculation above), and those are housed in the matrix prob-  
 415 cap. Our data simulator function `simSRC0.fn` will return the full 3-d array if  
 416 `array3d=TRUE` is specified in the function call. To recover the 2-d matrix from  
 417 the 3-d array, and subset the 3-d array to individuals that were captured, we  
 418 do this:

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

## 422 1.5 Fitting an SCR Model in BUGS

423 Clearly if we somehow knew the value of  $N$  then we could fit this model directly  
 424 because, in that case, it is a special kind of logistic regression model - one with  
 425 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.

```

431 data<-simSCR0.fn(discard0=FALSE,sd=2013)
432 y<-data$Y
433 traplocs<-data$traplocs
434 nind<-nrow(y)
435 X<-data$traplocs
436 J<-nrow(X)
437 y<-rbind(y,matrix(0,nrow=(100-nrow(y)),ncol=J ) )
438 Xl<-data$xlim[1]
439 Yl<-data$ylim[1]
440 Xu<-data$xlim[2]
441 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`.

```

452 cat("
453 model {
454   alpha0~dnorm(0,.1)
455   logit(p0)<- alpha0
456   beta~dnorm(0,.1)
457   for(i in 1:N){
458     s[i,1]~dunif(Xl,Xu)
459     s[i,2]~dunif(Yl,Yu)
460     for(j in 1:J){
461       d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
462       y[i,j] ~ dbin(p[i,j],K)
463       p[i,j]<- p0*exp(- beta*d[i,j]*d[i,j])
464     }
465   }
466 }
467 "
468 ",file = "SCR0a.txt")

```

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

```

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),beta=runif(1,1,2),s=sst)
}

library("R2WinBUGS")
parameters <- c("alpha0","beta")
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 Web Supplement. 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 **beta**). 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
beta	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).

513  
 514 DIC info (using the rule,  $pD = \bar{D} - \hat{D}$ )  
 515  $pD = -138.8$  and  $DIC = 154.0$   
 516 DIC is an estimate of expected predictive error (lower deviance is better).

517 We know the data were generated with  $\alpha_0 = -2.5$  and  $\beta = -2$ . The  
 518 estimates look reasonably close to those data-generating values and we probably  
 519 feel pretty good about the performance of the Bayesian analysis and MCMC  
 520 algorithm that WinBUGS cooked-up based on our sample size of 1 data set. It  
 521 is worth noting that the Rhat statistics indicate reasonable convergence but, as  
 522 a practical matter, we might choose to run the MCMC algorithm for additional  
 523 time to bring these closer to 1.0 and to increase the effective posterior sample  
 524 size ( $n_{\text{eff}}$ ). Other summary output includes “deviance” and related things  
 525 including the deviance information criterion (DIC). We discuss these things in  
 526 chapter XXXX model selection XXXX.

## 527 1.6 Unknown N

528 In all real applications N is unknown and that fact is kind of an important  
 529 feature of the capture-recapture problem! We handled this important issue  
 530 in Chapter 3 using the method of data augmentation which we apply here to  
 531 achieve a realistic analysis of Model SCR0. As with the basic closed population  
 532 models considered previously, we formulate the problem here by augmenting  
 533 our observed data set with a number of “all zero” encounter histories - what we  
 534 referred to in Chapter 3 as potential individuals. If  $n$  is the number of observed  
 535 individuals, then let  $M - n$  be the number of potential individuals in the data set.  
 536 For the basic  $y[i, j]$  data structure (individuals x traps encounter frequencies)  
 537 we simply add additional rows of “all 0” observations to that data set. This  
 538 is because such “individuals” are unobserved, and therefore necessarily have  
 539  $y[i, j] = 0$  for all  $j$ . A data set, say with 4 traps and 6 individuals, augmented  
 540 with 4 pseudo-individuals therefore might look like this:

	trap1	trap2	trap3	trap4
541 [1,]	1	0	0	0
542 [2,]	0	2	0	0
543 [3,]	0	0	0	1
544 [4,]	0	1	0	0
545 [5,]	0	0	1	1
546 [6,]	1	0	1	0
547 [7,]	0	0	0	0
548 [8,]	0	0	0	0
549 [9,]	0	0	0	0
550 [10,]	0	0	0	0
551 [11,]	0	0	0	0

552 We typically have more than 4 traps and, if we’re fortunate, many more  
 553 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 Chapter 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. 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  $D = N/area(S)$ . For our simulator, we’re using an  $8 \times 8$  state-space and thus we will compute  $D$  as  $D = N/64$ .

### 1.6.1 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  $\psi$ .

```
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("

```



```

597 model {
598   alpha0~dnorm(0,.1)
599   logit(p0)<- alpha0
600   beta~dnorm(0,.1)
601   psi~dunif(0,1)
602
603   for(i in 1:M){
604     z[i] ~ dbern(psi)
605     s[i,1]~dunif(Xl,Xu)
606     s[i,2]~dunif(Yl,Yu)
607     for(j in 1:J){
608       d[i,j]<- pow(pow(s[i,1]-X[j,1],2) + pow(s[i,2]-X[j,2],2),0.5)
609       y[i,j] ~ dbin(p[i,j],K)
610       p[i,j]<- z[i]*p0*exp(- beta*d[i,j]*d[i,j])
611     }
612   }
613   N<-sum(z[])
614   D<-N/64
615 }
616 ",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  $s_i$  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.

```

623
624 ## Data augmentation stuff
625 M<-200
626 y<-rbind(y,matrix(0,nrow=M-nind,ncol=ncol(y)))
627 z<-c(rep(1,nind),rep(0,M-nind))
628
629 sst<-cbind(runif(M,Xl,Xu),runif(M,Yl,Yu)) # starting values for s
630 for(i in 1:nind){
631   if(sum(y[i,])==0) next
632   sst[i,1]<- mean( X[y[i,]>0,1] )
633   sst[i,2]<- mean( X[y[i,]>0,2] )
634 }
635 data <- list (y=y,X=X,K=K,M=M,J=J,Xl=Xl,Yl=Yl,Xu=Xu,Yu=Yu)
636 inits <- function(){
637   list (alpha0=rnorm(1,-4,.4),beta=runif(1,1,2),s=sst,z=z)
638 }
639
640 library("R2WinBUGS")
641 parameters <- c("alpha0","beta","N")

```

```

642 nthin<-1
643 nc<-3
644 nb<-1000
645 ni<-2000
646 out <- bugs (data, inits, parameters, "SCR0a.txt", n.thin=nthin,n.chains=nc,
647   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  $N(S)$  is a derived parameter, being computed by summing up all of the data augmentation variables  $z_i$  (as we've done previously); (4) Density,  $D$ , is also a derived parameter. Summarizing the output from WinBUGS produces:

```

656 > print(out1,digits=2)
657 Inference for Bugs model at "SCR0a.txt", fit using WinBUGS,
658   3 chains, each with 2000 iterations (first 1000 discarded)
659   n.sims = 3000 iterations saved
660
660      mean      sd   2.5%   25%   50%   75%  97.5% Rhat n.eff
661 alpha0   -2.57  0.23  -3.04  -2.72  -2.56  -2.41  -2.15  1.01   320
662 beta      2.46  0.42   1.63   2.16   2.46   2.73   3.33  1.02   120
663 N       113.62 15.73  86.00 102.00 113.00 124.00 147.00 1.01   260
664 D         1.78  0.25   1.34   1.59   1.77   1.94   2.30  1.01   260
665 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,  $pD = \text{var}(\text{deviance})/2$ )

$pD = 279.9$  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 (*alpha0* and *beta*) 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 on the Web Supplement. 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  $R_{\text{hat}}$ ) 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 XYZ.

```

> 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).

```

## 1.6.2 Use of other BUGS engines: JAGS

There are two other popular BUGS engines: 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 which 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 (???) 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.

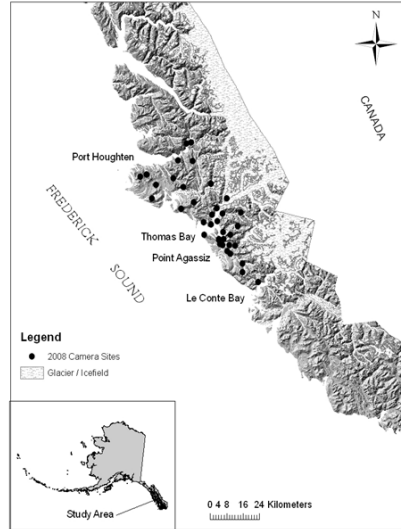


Figure 1.3: Wolverine locations.

## 1.7 Case Study: Wolverine Camera Trapping Study

We provide an analysis here of A. Magoun’s wolverine data (Magoun et al., 2011; ?). The study took place in SE Alaska (Figure 1.3) 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. XXXX) 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. xxxx accordingly. Our treatment of the data here is based on the analysis of ?.

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 efficient file format which is easily manipulated and also used as the input file format in our custom R script and SPACECAP (this is the “spacecap flat format” see section XYZ above). 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 ID. The first 10 rows of this matrix are as follows:

```
> wcaps
      trapid individual sample
[1,]      1           2     127
```

750	[2,]	1	2	128
751	[3,]	1	2	129
752	[4,]	1	18	130
753	[5,]	2	3	106
754	[6,]	2	18	104
755	[7,]	5	5	73
756	[8,]	5	5	89
757	[9,]	6	18	117
758	[10,]	6	18	118
759				

760 This “encounter data file” contains 1 row for each unique individual/trap en-  
 761 counter, and 3 variables (columns): trapid is an integer that runs from 1:ntraps,  
 762 individual runs from 1:nind and sample runs from 1:nperiods. Often (as the case  
 763 here) “samples” will correspond to daily sample intervals. The variable trapid  
 764 will have to correspond to the row of a matrix containing the trap coordinates  
 765 - a file called traplocs.csv on the web supplement.

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

775 The “encounter data file” wcaps.csv exists on the Web Supplement as a  
 776 CSV file that people can read into R and do some basic summary statistics  
 777 on. For our purposes we need to convert these data into the “individual x  
 778 trap” array of binary encounter frequencies, although more general models might  
 779 require an encounter-history formulation of the model which requires a full 3-  
 780 d array. To obtain our nind x ntrap encounter frequency matrix, we do this  
 781 the hard way by first converting the encounter data file into a 3-d array and  
 782 then summarize to trap totals. We have a handy function “SCR23darray.fn”  
 783 which takes the compact encounter data file with optional arguments ntraps  
 784 and nperiods, and converts it to a 3-d array, and then we use the R function  
 785 “apply” to summarize over the “sample period” dimension (by convention here,  
 786 this is the 2nd dimension):

```

787 SCR23darray.fn <- function(caps,ntraps=NULL,nperiods=NULL){
788   nind<-max(caps[,2])
789   if(is.null(ntraps)) ntraps<-max(caps[,1])
790   if(is.null(nperiods)) nperiods<- max(caps[,3])
791
792   y<-array(0,c(nind,nperiods,ntraps))
793   tmp<-cbind(caps[,2],caps[,3],caps[,1])

```

```

794 y[tmp]<-1
795 y
796 }
797
798 # for the wolverine data do this:
799
800 Y3d <-SCR23darray.fn(wcaps,ntraps=37,nperiods=165)
801 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” on the Web Supplement) 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 run a model in WinBUGS we do this:

```

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

Table 1.1: 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	

### 1.7.1 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)  $y[i, j] | s[i] \sim \text{Binomial}(K, z[i] * p[i, j])$
- (2)  $p[i, j] = p0 * \exp(-\text{beta} * ||s[i] - x[j]||^2)$
- (3)  $s[i] \sim \text{Uniform}(S)$
- (4)  $z[i] \sim \text{Bern}(\text{psi})$

We assume customary flat priors on the structural (hyper-) parameters of the model, alpha, beta, and psi. It remains to define the state-space S. For this, we overlaid the trap array (Fig. XXX.YYY) on a rectangular region 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 10 km, producing a rectangular state-space of dimension  $9.88 \times 10.5$  units ( $area = 10374 km * km$ ) 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 maybe should be provided...?]. 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,

```



```

905 3 chains, each with 12000 iterations (first 2000 discarded)
906 n.sims = 30000 iterations saved
907      mean    sd    2.5%    25%    50%    75%    97.5% Rhat n.eff
908 psi      0.19 0.04  0.11  0.16  0.19  0.22  0.29 1.00   790
909 sigma    1.80 0.34  1.30  1.56  1.75  1.98  2.59 1.01   400
910 p0       0.03 0.00  0.02  0.03  0.03  0.03  0.04 1.00  2800
911 N        56.66 11.47 37.00 48.00 56.00 64.00 82.00 1.00   570
912 D         4.53 0.92  2.96  3.84  4.48  5.11  6.55 1.00   570
913 beta     0.17 0.06  0.07  0.13  0.16  0.20  0.30 1.01   400
914 deviance 444.75 11.87 423.60 436.40 444.00 452.30 469.80 1.00 24000
915
916 Buffer = 30 km
917 > print(out4$out,digits=2)
918 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
919 3 chains, each with 12000 iterations (first 2000 discarded)
920 n.sims = 30000 iterations saved
921      mean    sd    2.5%    25%    50%    75%    97.5% Rhat n.eff
922 psi      0.23 0.05  0.14  0.19  0.22  0.26  0.34 1.00  1500
923 sigma    1.79 0.34  1.29  1.55  1.73  1.97  2.58 1.01   560
924 p0       0.03 0.00  0.02  0.03  0.03  0.03  0.04 1.00 30000
925 N        67.39 14.12 43.00 57.00 66.00 76.00 98.00 1.00  1200
926 D         4.54 0.95  2.90  3.84  4.44  5.12  6.60 1.00  1200
927 beta     0.17 0.06  0.07  0.13  0.17  0.21  0.30 1.01   560
928 deviance 444.58 11.83 423.60 436.40 443.80 452.20 469.90 1.00  4700
929
930 Buffer = 45 km
931 > print(out7$out,digits=2)
932 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
933 3 chains, each with 12000 iterations (first 2000 discarded)
934 n.sims = 30000 iterations saved
935      mean    sd    2.5%    25%    50%    75%    97.5% Rhat n.eff
936 psi      0.36 0.08  0.21  0.30  0.35  0.41  0.53  1  5000
937 sigma    1.78 0.34  1.29  1.55  1.72  1.95  2.60  1   850
938 p0       0.03 0.00  0.02  0.03  0.03  0.03  0.04  1  3600
939 N       106.57 23.34 67.00 90.00 104.00 121.00 157.00  1  3400
940 D         4.62 1.01  2.90  3.90  4.51  5.25  6.81  1  3400
941 beta     0.17 0.06  0.07  0.13  0.17  0.21  0.30  1   850
942 deviance 444.80 11.84 423.60 436.40 444.10 452.30 470.00  1 30000
943
944 Buffer = 50 km
945 > print(out8$out,digits=2)
946 Inference for Bugs model at "modelfile.txt", fit using WinBUGS,
947 3 chains, each with 12000 iterations (first 2000 discarded)
948 n.sims = 30000 iterations saved
949      mean    sd    2.5%    25%    50%    75%    97.5% Rhat n.eff
950 psi      0.40 0.09  0.23  0.33  0.39  0.45  0.60 1.01  1300

```

951	<b>sigma</b>	1.82	0.48	1.30	1.56	1.74	1.97	2.68	1.05	200
952	<b>p0</b>	0.03	0.00	0.02	0.03	0.03	0.03	0.04	1.00	5800
953	<b>N</b>	118.47	26.81	71.00	100.00	117.00	135.00	176.00	1.01	1200
954	<b>D</b>	4.52	1.02	2.71	3.82	4.46	5.15	6.72	1.01	1200
955	<b>beta</b>	0.17	0.06	0.07	0.13	0.17	0.21	0.30	1.05	200
956	<b>deviance</b>	444.84	11.90	423.90	436.50	444.10	452.20	470.30	1.00	500

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

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

## 971 1.7.2 Conclusion of Analysis

972 Our point estimate of wolverine density from this study of approximately 4.5  
 973 individuals/1000 km\*km and a 95% posterior interval is around [2.7, 6.3] indi-  
 974 cating that density is estimated imprecisely, obviously due to the low sample  
 975 size (n=21 individuals!). This seems to be a basic feature of carnivore studies.

976 It is worth thinking about this model, and these estimates, computed under  
 977 a rectangular state space roughly centered over the trapping array (Figure  
 978 XXXX). Does it make sense to define the state-space to include, for example,  
 979 ocean? What are the possible consequences of this? What can we do about it?  
 980 There's no reason at all that the state space has to be a regular polygon - we  
 981 defined it as such here strictly for convenience and for ease of implementation  
 982 in WinBUGS where it enables us to specify the prior for the activity centers  
 983 as uniform priors for each coordinate. While it would be possible to define a  
 984 more realistic state-space using some general polygon, it might take some effort  
 985 to implement that in the BUGS language (See chapter X.Y for example of a  
 986 simple case). Alternatively, we recommend using a discrete representation of  
 987 the state-space - i.e., approximate S by a grid of G points. We discuss this in  
 988 the following section.

## 989 1.8 Constructing Density Maps

990 One of the most useful aspects of SCR models is that they are parameterized  
 991 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 50%. 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 process  $(\mathbf{s}[1], \dots, \mathbf{s}[N])$ . In the present context, the number of individuals living in any well-defined polygon is a derived parameter. Specifically, let  $B(x)$  indicate a box centered at  $x$  then  $N(x) = \sum_i I(\mathbf{s}[i] \in B(x))$  is the population size of box  $B(x)$ , and  $D(x) = N(x)/|B(x)|$  is the local density. These are just “derived parameters” (see Chapt. 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  $m$ , we only tabulate those activity centers which correspond to individuals in the sampled population. i.e., for which the data augmentation variable  $z_i = 1$ . In this case, we take all of the output for MCMC iterations  $m = 1, 2, \dots$ , and compute this summary:

$$N(x, m) = \sum_{z[i, m]=1} I(\mathbf{s}[i, m] \in B(x))$$

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

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

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

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

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

**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)
```

1029 **Step 5:** Use the `image()` command to display the resulting matrix.

1030 `image(xg,yg,Dn/nrow(z),col=terrain.colors(10))`

1031 Praise the Lord! This map is somewhat useful or at least it looks pretty and  
1032 will facilitate the publication of your papers.

1033 It is worth emphasizing here that density maps will not usually appear uni-  
1034 form despite that we have assumed that activity centers are uniformly dis-  
1035 tributed. This is because the observed encounters of individuals provide direct  
1036 information about the location of the  $i = 1, 2, \dots, n$  activity centers and thus  
1037 their “estimated” locations will be affected by the observations. In a limiting  
1038 sense, were we to sample space intensely enough, every individual would be  
1039 captured a number of times and we would have considerable information about  
1040 all  $N$  point locations. Consequently, the uniform prior would have almost no  
1041 influence at all on the estimated density surface in this limiting situation. Thus,  
1042 in practice, the influence of the uniformity assumption increases as the fraction  
1043 of the population encountered decreases.

1044 **On the non-intuitiveness of `image()`** - The R function `image()` is not a  
1045 very intuitive function - it plots  $M[1,1]$  in the lower left corner which might be  
1046 confusing. If you want  $M[]$  to be plotted “as you look at it” then  $M[1,1]$  should  
1047 be in the upper left corner. We have a function `rot()` which does that. If you do  
1048 `image(rot(M))` then it puts it on the monitor as if it was a map you were looking  
1049 at. You can always specify the x and y- labels explicitly as we did above.

1050 **Spatial dot plots** - Now here is a cruder version based on the “spatial  
1051 dot map” function “`spatial.plot`”. The useful functions in R are `image()` and  
1052 `image.scale()` which is a function we grabbed off of the web somewhere. Use of  
1053 this function requires arguments of point locations and the resulting value to be  
1054 displayed. The function is defined and applied as follows:

```
1055 spatial.plot<- function(x,y){
1056   nc<-as.numeric(cut(y,20))
1057   plot(x,pch=" ")
1058   points(x,pch=20,col=topo.colors(20)[nc],cex=2)
1059   image.scale(y,col=topo.colors(20))
1060 }
1061 # To execute the function do this:
1062 spatial.plot(cbind(xg,yg), Dn/nrow(z))
```

### 1063 1.8.1 Example: Wolverine density map.

1064 We used the posterior output from the wolverine model fitted previous to com-  
1065 pute a relatively coarse version of a density map, using a 10 x 10 grid (Figure  
1066 XXX.YYY) and using a 30 x 30 grid (Figure XXYYZZ). In both cases, the den-  
1067 sity is expressed “per pixel”, and hence the differing scales<sup>3</sup>. A couple of things  
1068 are noteworthy: First is that as we move away from “where the data live” - away

---

<sup>3</sup>Andy needs to recompute these in a standardized way

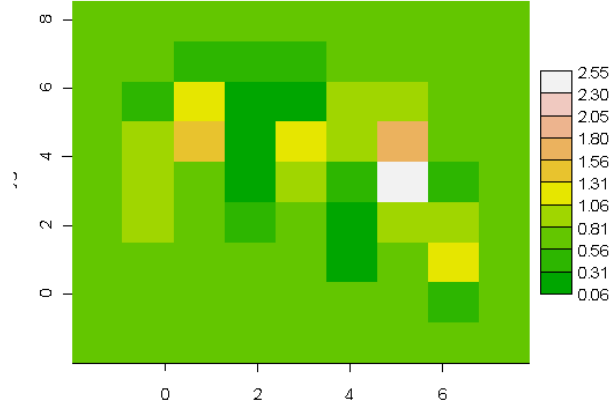


Figure 1.4: Needs a caption

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. 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. 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.

## 1.9 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 this will not always be a reasonable assumption. In chapter XYZ 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 might choose to

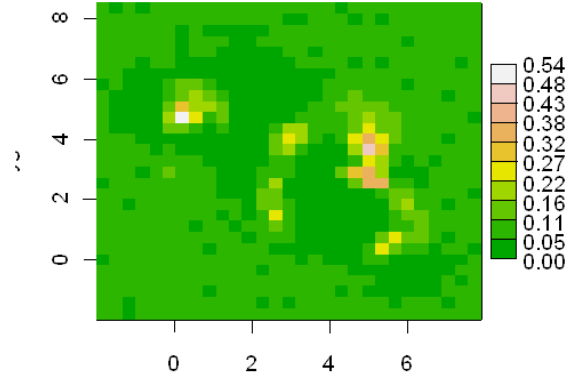


Figure 1.5: Needs a caption

1092 remove the non-habitat from the state-space and assume uniformity of the activ-  
 1093 ity centers over the remaining portions of the state-space judged to be habitat.  
 1094 There are two ways to approach this: We can use a regular grid of points to rep-  
 1095 resent the state-space, i.e.,  $s_1, \dots, s_G$  and assign a discrete uniform distribution  
 1096 to each individual's activity center, or we can retain the continuous formulation  
 1097 of the state-space but use basic polygon operations to induce constraints on the  
 1098 state-space (Chapter MCMC and also Appendix XYZ). We focus here on the  
 1099 formulation of our basic SCR model in terms of a discrete state-space. Use of a  
 1100 discrete state-space can be computationally expensive in WinBUGS. That said,  
 1101 it isn't too difficult to do the MCMC calculations in R which we discuss briefly  
 1102 in Chapter XYZ. The R package SPACECAP (Gopalaswamy, 2011) arose from the  
 1103 R implementation developed for the application in Royle et al. (2009) (Ecol-  
 1104 ogy paper). As we will see in chapter 5, we must prescribe the state-space by a  
 1105 discrete mesh of points in order to do integrated likelihood and so if we are using  
 1106 a discrete state-space this can be accommodated directly for obtaining MLEs.  
 1107 While clipping out non-habitat seems like a good idea, its not obvious that we  
 1108 accomplish any biologically reasonable objective by doing so. We might prefer  
 1109 to do it when non-habitat represents a clear-cut restriction on the state-space  
 1110 such as a reserve boundary or a lake, ocean or river. It makes sense in those  
 1111 situations. Unfortunately, having the capability to do this also causes people  
 1112 to start defining "habitat" vs. "non-habitat" based on their understanding of  
 1113 the system whereas it can't be known whether the animal being studied has the  
 1114 same understanding. Moreover, differentiating of the landscape by habitat or  
 1115 habitat quality probably affects the geometry and morphology of home ranges  
 1116 much more than the plausible locations of activity centers. That is, a home  
 1117 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 Section XYZ).

### 1.9.1 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<sup>4</sup>. Why is this? We can think about this as doing numerical integration.... We don't need a huge amount of support points to evaluate the integral to a reasonable level of accuracy. We demonstrate this here by reanalyzing simulated data for different approximations to the state-space. We create a new version of the data simulator called `simSCR0discrete.fn`.... As noted in section 1.3.4 above, we can think about SCR models as a type of finite mixture (Norris III and Pollock, 1996; Pledger, 2000) with direct information about which "group" individuals belong to. In these finite mixture models we typically find that only 1 or a very small number of groups can explain really high levels of heterogeneity. We therefore expect a similar effect in SCR models when we discretize the state-space.

### 1.9.2 Analysis of the wolverine camera trapping data

We reanalyzed the wolverine data using grids with points spaced by 2, 4 and 8 km (Fig XYZ). What was the effect of this? We also provide a second analysis of the data in which we used a discrete representation of the state-space, but then deleted points that were not over land. For that approach we used 2 grid resolutions (2 km and 8 km) before clipping out unsuitable points. 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 not if there is a strong influence on estimated density.

### 1.9.3 SCR models as multi-state models

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 XYZ. While we

---

<sup>4</sup>although the geometry of the state-space is likely to have a big effect

1156 invoke a discrete state-space artificially, by gridding the underlying continuous  
 1157 state-space, sometimes the state-space is more naturally discrete. Consider a  
 1158 situation in which discrete patches of habitat are searched using some method  
 1159 and it might be convenient (or occur inadvertently) to associate samples to the  
 1160 patch level instead of recording observation locations. In this case we might  
 1161 use a model  $s[i] \sim dcat(probs[])$  where  $probs[]$  are the probabilities that an in-  
 1162 dividual inhabits a particular patch. We consider such a case study in Chapter  
 1163 XXPoissonXXX from Mollet et al. (2012) who obtained a population size esti-  
 1164 mate of a large grouse species known as the capracaille. Forest patches were  
 1165 searched for scat which was identified to individual by DNA analysis. Even  
 1166 when space is *not* naturally discrete, measurements are often made at a fairly  
 1167 coarse grain (e.g., meters or tens of meters along a stream), or associated with  
 1168 spatial quadrats for scat searches. Even so, we could approximate any contin-  
 1169 uous measurement using a discrete state-space, and therefore apply multi-state  
 1170 models directly to any SCR problem.

## 1171 1.10 Summary and Outlook

1172 A point we tried to emphasize in this chapter is that the basic SCR model is not  
 1173 much more than an ordinary capture-recapture model for closed populations –  
 1174 it is simply that model augmented with a set of “individual effects”,  $s[i]$ , which  
 1175 relate some sense of individual location to encounter probability. SCR models  
 1176 are therefore a type of individual covariate model (as introduced in Chapter  
 1177 3) - but with imperfect information about the individual covariate. In other  
 1178 words, GLMM type models when N is known or, when N is unknown, they are  
 1179 zero-inflated GLMMs (see Royle (2006)). These models are really quite easy to  
 1180 analyze by likelihood methods, based on the integrated likelihood, and they are  
 1181 also very easy to analyze using existing MCMC black boxes such as WinBUGS or  
 1182 JAGS and possibly other packages. We will consider likelihood analysis of such  
 1183 models sparingly in this book (but see Chapter XYZ) because our emphasis is on  
 1184 Bayesian analysis. Formal consideration of the collection of individual locations  
 1185 ( $s[1], \dots, s[N]$ ) in the model is fundamental to all of the models considered in this  
 1186 book. In statistical terminology, we think of the collection of points  $\{s_i\}$  as a  
 1187 realization of a point process and part of the promise, and ongoing challenge, of  
 1188 SCR models is to develop SCR models based on interesting point process mod-  
 1189 els. Here we considered the simplest possible point process model - the points  
 1190 are independent and uniformly (“randomly”) distributed over space. Despite  
 1191 the simplicity of this assumption, it should suffice in many applications of SCR  
 1192 models although we do address generalizations of this model in later chapters.  
 1193 Moreover, even though the *prior* distribution on the point locations is uniform,  
 1194 the realized pattern may deviate markedly from uniformity as the observed en-  
 1195 counter data provide information to impart deviations from uniformity. Thus,  
 1196 the estimated density map will typically appear distinctly non-uniform. As a  
 1197 general rule, information in the data will govern estimates of individual point  
 1198 locations so even fairly complex patterns of non-independence or non-uniformity



will appear in the data. 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. 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” which we develop in Chapter XYZ. 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? That’s a good question. The main reason is that these conventional methods are predicated on models that are blatant misspecifications of the observation and ecological process - they are wrong! Or perhaps more charitably, they 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. And, “density” is not a parameter of those models because density has no meaning absent an explicit representation of space. 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. We consider technical details of Bayesian and maximum likelihood estimation in the following chapter, and then extensions to include covariates that vary by individual, trap, or over time (chapter XXX.YYY).



# Bibliography

- 1229
- 1230 Arnason (1973), “Missing,” *Missing*, Missing, Missing.
- 1231 — (1974), “Missing,” *Missing*, Missing, Missing.
- 1232 Borchers, D. and Efford, M. (2008), “Spatially explicit maximum likelihood  
1233 methods for capture–recapture studies,” *Biometrics*, 64, 377–385.
- 1234 Efford, M. (2004), “Density estimation in live-trapping studies,” *Oikos*, 106,  
1235 598–610.
- 1236 Gardner, B., Royle, J., Wegan, M., Rainbolt, R., and Curtis, P. (2010), “Esti-  
1237 mating black bear density using DNA data from hair snares,” *The Journal of*  
1238 *Wildlife Management*, 74, 318–325.
- 1239 Gopalaswamy (2011), “Missing,” *Missing*, missing.
- 1240 Hestbeck (1991), “Missing,” *Missing*, Missing, Missing.
- 1241 Illian (2008), “Missing,” *Missing*, Missing.
- 1242 Kéry, M., Gardner, B., Stoeckle, T., Weber, D., and Royle, J. A. (2010), “Use  
1243 of Spatial Capture-Recapture Modeling and DNA Data to Estimate Densities  
1244 of Elusive Animals,” *Conservation Biology*, 25, 356–364.
- 1245 Kery, M. and Schaub, M. (2011), *Bayesian Population Analysis Using WinBugs*,  
1246 Academic Press.
- 1247 Link, W. A. (2003), “Missing,” *missing*, missing.
- 1248 Magoun, A. J., Long, C. D., Schwartz, M. K., Pilgrim, K. L., Lowell, R. E.,  
1249 and Valkenburg, P. (2011), “Integrating motion-detection cameras and hair  
1250 snags for wolverine identification,” *The Journal of Wildlife Management*, 75,  
1251 731–739.
- 1252 Mollet, P., Kery, M., Gardner, B., Pasinelli, G., and A, R. J. (2012), “Popu-  
1253 lation size estimation for capercaillie (*Tetrao urogallus* L.) using DNA-based  
1254 individual recognition and spatial capture-recapture models,” *missing*, miss-  
1255 ing, missing.

- 1256 Norris III, J. L. and Pollock, K. H. (1996), “Nonparametric MLE under two  
1257 closed capture-recapture models with heterogeneity,” *Biometrics*, 639–649.
- 1258 Pledger, S. (2000), “Unified maximum likelihood estimates for closed capture-  
1259 recapture models using mixtures,” *Biometrics*, 434–442.
- 1260 Plummer, M. (2003), “JAGS: A program for analysis of Bayesian graphical mod-  
1261 els using Gibbs sampling,” in *Proceedings of the 3rd International Workshop*  
1262 *on Distributed Statistical Computing (DSC 2003)*. March, pp. 20–22.
- 1263 — (2009), “rjags: Bayesian graphical models using mcmc. R package version  
1264 1.0. 3-12,” .
- 1265 Royle, J. (2006), “Site occupancy models with heterogeneous detection proba-  
1266 bilities,” *Biometrics*, 62, 97–102.
- 1267 Royle, J., Karanth, K., Gopalaswamy, A., and Kumar, N. (2009), “Bayesian  
1268 inference in camera trapping studies for a class of spatial capture-recapture  
1269 models,” *Ecology*, 90, 3233–3244.
- 1270 Royle, J. A. and Young, K. V. (2008), “A Hierarchical Model For Spatial  
1271 Capture-Recapture Data,” *Ecology*, 89, 2281–2289.
- 1272 Thomas, A., O’Hara, B., Ligges, U., and Sturtz, S. (2006), “Making BUGS  
1273 Open,” *R News*, 6, 12–17.