Lab 10 – Spatially explicit capture-recapture using the 'secr' package

This lab and handout was originally prepared by former WFCB PhD student John Clare

General (tangential) background: closed-capture shortcomings

Although capture-recapture modeling to estimate abundance in a closed population has been around for more than 100 years (see Lincoln, Peterson), modern closed capture-recapture modeling to estimate abundance was developed more recently than open models of the J-S or C-J-S variety. As we saw during week 7 on closed capture models, Otis et al. (1978) defined several potential detection models: a null model (M_0) in which all individuals are equally detectable across the entire survey, a model in which individual detection probability changes after initial detection (M_B), a model in which detection probability changes over time (M_T), a model in which individuals have different intrinsic detection probabilities (M_H), and different combinations of these factors. Of these models, M_H has seen most the most focus, as individual heterogeneity in detection probability is both ecologically reasonable, very common within data-sets, and surprisingly difficult to model. Conditional likelihood models (Huggins 1991) can be implemented in MARK and RMark to allow detection probability to vary according to individual covariates. The cost is that Huggins models condition N-hat out of the likelihood, making it impossible to model variation in abundance. There wasn't even a full-likelihood method for accommodating heterogeneity until the development of finite mixture models (Norris and Pollock et al. 1996, Pledger et al. 2000, 'heterogeneity' in MARK) [as an aside, although Otis et al. used a jack-knife procedure to estimate abundance when heterogeneity was the selected model, Ken Burnham had conceptualized a beta-binomial random effect model for heterogeneity in 1972]. Although finite mixtures, beta-binomial random effects, or logistic normal random effects (Dorazio and Royle 2003) are all valid means of accommodating heterogeneity, it is impossible to select between these models, and they may provide vastly different estimates (Link 2003).

So, figuring out how to best model heterogeneity is one issue in capture-recapture. Assuming abundance is the parameter you're after, the next biggest conceptual problem in closed-capture development has been figuring out ways to model variation in abundance and detection using ecologically plausible factors (MARK apparently allows this, but it is a tedious process). This is exactly what hierarchical models are good at doing. Binomial N-mixture models (Royle 2004a) allow site and visit-specific variation in counts of unmarked animals, but because there is no individual encounter record, all animals are assumed equally detectable. Multinomial N-mixture models (Royle 2004b) represent an extension that incorporates individual encounter histories, the type you might collect with capture-recapture data, a removal study, a single-visit point count following a double-observer protocol, a single visit point count in which time of detection (or 'removal' from consideration) is recorded, etc. The site-specific encounter histories look like a matrix with rows representing sites, columns representing encounter histories (e.g. "10, 01, 11"), and the cell values are counts of individuals at sites that shared a specific encounter history. You can easily fit these models using the 'unmarked' package under a multinomial Poisson or generalized multinomial mixture functions, and compare or combine the individual based detection models of Otis with models allowing detection to vary based upon site or time specific covariates. These multinomial N-mixture models are really efficient for sampling

designs that feature a lot of independent sites (e.g., point counts), something difficult to achieve in most capture-recapture studies. For example, a small mammal trapping grid and a single visit to listen to birds would each count as a site, even though the effort is hardly equitable: recall that the number of sites generally dictates the number of covariates that you might use to model variance in abundance. Richard Chandler has written a nice vignette describing the capabilities of 'unmarked' for the analysis of capture-recapture data.

But wait, there is yet one more problem: abundance is only a valuable absolute metric if you know what area an abundance estimate corresponds with…i.e., density. For most capture recapture data, the effective sampling area is not obviously defined. There are two general ways to approach this challenge that tend to depend upon your sampling scheme, but share one fundamental concept: animals become more difficult to detect if they are further away. The first technique for evaluating the effective sampling area (and estimating density) is for a sampling scheme in which individuals are fixed at points in space but detectable over large areas, and the second technique is for sampling problems in which detection is fixed at specific points in space, but individuals range across broader areas. The first technique is distance sampling, which relies on a hazard function of some form to describe how detectability of individuals declines as they are further away from you or an acoustic recorder or whatever.

Spatial Capture-Recapture (or spatially explicit capture-recapture)

The second technique is spatially explicit capture-recapture (either SECR or SCR, depending upon your preference), introduced more or less simultaneously by Borchers and Efford, Royle and Young, and Gardner et al. in 2008. The general sampling concept is this: you have traps or detectors that are fixed in space, and not independent (as sites would be assumed to be for distance sampling). Animals *i* can move between traps *j*, but are assumed to have an activity center (s_i) that is fixed over the duration of the survey (either a home range center, or perhaps a temporary point of gravity). Detection probability is modeled as a hazard function (exactly like distance sampling), such that station-specific (*j*) detection probability *gij* decreases as a station is further away from an individual's activity center (s_i). Note that this means that an individual's detection probability is changing across space; that what we think of as *p* over a time period *k* really becomes a summation of *g's*: in other words, the detection probability for an individual during a sampling occasion is the summation of the station-specific detection probabilities for that individual at all particular points in space where we have traps. Like distance sampling, estimates of the distance parameters allow estimation of the effective sampling area. But unlike distance sampling, detection probability is not 1 at a distance of 0: even though an individual might be most easily detected at its activity center, it is obviously not always present at that location.

The easiest way to describe the guts of the density process here is to take a look at the hierarchical model itself. Again, we index individuals as *i,* trap locations as *j*, and a sampling occasion as *k,* within an overall space we refer to as *S:.*

$$N \sim Poisson(\mu||S||)$$

Abundance is again considered the realization of a Poisson process (actually a spatial point process in this case). Parameter μ is equivalent to the intensity of this spatial point process, or the average number of activity centers s_i per some finite unit in continuous space. In other words, μ is basically population density, and the R library 'secr' refers to this parameter as 'D'. 'D' at its simplest is an intercept only model—which sounds like density is essentially fixed and homogenous in space, but really means that the random realization of activity centers at any given subsection are draws from a common distribution. The 'D' parameter, like all others we've worked with to date, can easily be modeled in a GLM fashion so that average density changes according to some covariate.

$$s_i \sim \text{Uniform}(S)$$

Within the simplest spatial capture-recapture models, we assume that any given individual has an equal and uniform probability of being located at any particular subsection within the overall space (S). This is analogous to saying that an individual encountered could live anywhere within the overall space. Although continuous space is no challenge for Bayesian estimation based on simulation, an MLE in continuous space would require multiple higher-level integrations. Thus, MLE generally requires discretizing the total space into grid cells because summation is easier than integration: this doesn't practically matter in most cases, and besides, all of the spatial information we would like to use to consider variation comes in some discrete form as a raster or vector. Within MLE analysis in 'secr', the location of individual activity centers is marginalized out of the likelihood, and indeed, the realization of N as the sum of activity centers is marginalized out, but can be kind of reconstructed.

If we take the product of μ and the number of subsections of equal size in S, we have total expected population size, with N itself being a random realization of the expectation. An alternative way to think of this is as N being the sum of all activity centers *s* that exist.

$$Y_{ijk} | \, s_i \sim \text{Bernoulli}(p_{ij}(||x_j - s_i||))$$

During some finite sampling occasion, the probability of detecting an individual at a specific trap is a Bernoulli outcome with probability dependent upon the Euclidean distance between the individual's activity center and the trap location. Like occupancy or N-mixture models, this can be easily summarized as a Binomial summation of a Bernoulli detection process across several locations or times locations. The encounter process could also be modeled as a count over an average duration, e.g., $\text{Poisson}(p_{ij}(||x_j - s_i||))$.

$$p_{ij} = g_0 \exp(-||\mathbf{x}_j - \mathbf{s}_i||^2 / (2\sigma^2))$$

The specific distance function that is used to define the Bernoulli probability above is a half-normal or bivariate normal distance function, where g_0 is the probability of detecting an individual directly at its activity center during an ocassion, and σ is a measure of how quickly detection probability declines moving away from its activity center ($2.45 * \sigma$ is an approximate measure of the maximum distance at which an individual could be detected under this half-normal function). But any distance related function might be appropriate: an organism that

patrols territory boundaries or uses its home range uniformly might be better modeled using different functions.

Whew! A couple notes: all of these formula can be altered depending upon the specific questions of interest, your sampling design, and your specific software choice. As examples, Bayesian analysis almost always treats abundance as a Binomial point process employing data augmentation, the location of activity centers (and μ) can be allowed to vary across space, encounter processes can be count based, and distance functions are incredibly varied. And extensions are innumerable. But conceptually, an SCR model is explicitly defining algebraic expressions for how individuals are located in space, and how they are encountered given how they use available space. There are a lot of mathematical assumptions that come with how these processes are defined. However, many of the same assumptions are implicit within closed capture models: requiring an explicit algebraic definition of space use allows comparison of different space use models, and forces an investigator to think a little more about the ecology of the species considered. We should note that density estimates from SCR models are fairly robust to many assumption violations (over or under-dispersion of individuals, misspecification of space-use models, etc.).

At any rate, here are some practical benefits of running an SCR analysis vs. a traditional closed capture approach:

1. Your sampling unit is a pixel of indeterminate size (see below), and the effective sample size (rows in a multinomial matrix or Bernoulli outcomes in a 3 dimensional array), and so:
    a. Given that individuals are (hopefully!) more prevalent than independent sites (the 'unmarked' sample unit), you have more degrees of freedom with which to model variation.
    b. This variation can be modeled at any scale you are interested in (and you can both extrapolate or interpolate population size trivially). Scale does not mean grain, because the distribution of activity centers within discrete subunits has to fit some sort of finite spatial point process.
2. Even the simplest encounter model explicitly incorporates heterogeneity into the detection process (via the location of individuals vs. traps) in a realistic fashion. However, the detection function can easily be modified to incorporate more realistic models of individual space use. In particular, models developed by Royle (2013*a*, 2013*b*) allow detection to vary as a function of least cost path distance vs. Euclidean distance, or resource selection. In other words, with enough captures, you can directly estimate movement cost or selection at the same time you estimate population density or size.

There are a couple of other important extensions (unfortunately, only some are covered by the software used in the lab, and none are dealt with in the lab itself). These include the use of spatially indexed/correlated raw counts to estimate density (spatial N-mixtures), consideration of linear habitats such as stream networks for estimation, and the extension of the closed capture-recapture process into the robust design. The last option offers incredible power but has limited

development: conceptualizing an animal as a point on a map makes it possible to directly consider how many there are, where they area, and via what demographic process they got there.

The 'Secr' package:

'Secr' was created by and maintained by Murray Efford, and the MLE techniques follows Borchers and Efford (2008). It has impressive capabilities, some very convenient functional features and is being actively supported, but suffers a few limitations (primarily, no ability to fit a truly Markovian open model, although no other MLE analysis has done so, either). In general, there are 3 stages to the input process:

1) Reading a traps object
   The trap input for secr is most conveniently read if each trap represents a row, with columns for the trapID (e.g, PR1), the x and y coordinates in meters (convenient for existing utms), and the activity history of the trap itself (say, if it were inoperable on the first occasion, something like "011111…"). You can later add trap specific covariates if desired using the addCovariates function, or include them preceded by a '/'. There are several trap types (or 'detectors'), and you must define the detector type. A brief description of each is as follows:
   a. "single" is basically a live trap that can only hold one animal. There actually is no likelihood that can accommodate this specific detection process, but it can be approximated.
   b. "multiple" is basically a live trap that can hold any number of animals. The key for single and multiple is that an animal can be detected only at one location during an occasion. The problem with single is that a captured animal precludes the detection of other animals at the same location. Borchers and Efford (2008) use a cumulative hazard function to address the 'trap filling up' problem for both 'single' and 'multiple' trap types.
   c. 'proximity': animals are not restricted by the trap, and can be detected at many places during an occasion
   d. 'count': as above, but animals can be detected several times at any one detector during an occasion. Use care with this option: are these detections really independent?
   e. Several 'polygon' or 'transect' options that might be generated using searches for scat or individuals within a 2d area or along a trail
   f. "signal strength': acoustic detectors.

2) Creating a capture history object
   Read.capthist is a function that combines your trap with an encounter history. The easiest format for the encounter history is to have columns designating identifiers for the 'session' (this could be a group, or it could represent different trap arrays, or different years), the individual animal (1, 2, or A, B, whatever), the occasions during which the animal was detected, and the locations of those detections (the trap name).

3) Reading a 'Mask"

Not technically required, the default mask is a blank space with some given pixel grain and spatial extent that can be defined within the model fitting function. A user-provided mask is basically a raster that might distinguish habitat from non-habitat (i.e. activity centers can only occur within certain parts of the overall space), or contain information about varied continuous or categorical covariates you might use to predict density (or define movement cost). It doesn't have to be a rectangle, but it does need to contain all of your trap locations. Beyond any specific covariates or what have you, there are 2 primary considerations when reading a custom mask: how far it extends beyond your trap locations, and what grain the pixels within the mask are. The first attribute of the mask is called a buffer: if it is too small, you are likely to vastly overestimate density. The gain or spacing of the mask can influence how well the spatial point process performs. It also obviously has some influence on the covariates you consider. I recommend using the 'secr' function mask.check (on a mask with no covariates) to check on things before creating a raster that you might input. Also keep in mind that the overall number of pixels is normally the biggest bottleneck for computing time.

The format for a mask is a text/csv/whatever with columns including x coordinates, y coordinates, and potentially any additional covariates associated with a given point.

The lab: For this week's lab the code, annotation, and a written description of each step are provided below. Code is bolded and italicized, annotation is bolded and preceded with some number of #, and the larger written description of each step is contained in standard font. I recommend that you proceed through the lab by copy/pasting code into RStudio and adding your own elaboration on the annotation as needed. Notice that this lab is based entirely on simulated data contained within the code and/or package, so there is no data file to download.

####We create a fake habitat-based mask and a population of animals within it

library(secr)

WLE650_mask<-make.mask(nx = 50, ny=50, spacing =10)

plot(WLE650_mask)

Here, you get a sense of what the mask actually is: each point represents a potential activity center location for the animals within our region. If you plot (dots=FALSE), you get something like a rasterized image. There are no covariates or additional information, so it looks like a bunch of grey dots, or a gray square.

set.seed(1420) ###so everybody gets the same results

WLE650_Hab<-randomHabitat(WLE650_mask, p=.6, A=.6, drop=F)

plot(WLE650_Hab, covariate="habitat", dots=F, col=c('brown', 'green'), breaks=2)

‘randomHabitat’ is a secr function that distributes habitat across a provided mask, p and A correspond to the proportion and clustering of habitat. We plot it, and see habitat classified as green, and non-habitat as brown. The drop argument corresponds to whether we remove non-habitat from the mask or not. Conceptually, you might only be interested in density within obvious habitat, or you might *a* priori constrain activity centers to only be located within certain habitat types (and not in urban areas, or something). If reading a mask from an input file, you can choose to clip out non-habitat beforehand.

Hab<-covariates(WLE650_Hab)

dens<-1.5*Hab

dens<-as.numeric(unlist(dens))

pop<-sim.popn(D=dens, core=WLE650_Hab,model2D="IHP", NDist='fixed')

pop

plot(pop, add=T)

We have simulated a population with a density of 1.5 individuals/ha in habitat, and 0 in non-habitat. Our entire area of inference is 25 hectares, so on average, we should expect 25*.6(the proportion of habitat)*1.5 =22.5 individuals in an area with these characteristics. The finite population within our particular sample ends up being 23. It is possible to hard-code simulated spaces, individuals, etc. within R (and this is better for understanding how the matrices are set up for this sort of problem), but the internal simulation functions are easy, and save you the trouble of manipulating objects to fit ‘secr’s strange input style.

Traps<-make.grid(nx=6, ny=6, spacex = 65 ,spacey= 50, detector='multi', originxy=c(100,100))

plot(Traps, add=T)

Here are our make-believe live traps (something like a giant pitfall that can capture several individual animals and restrict their movement). We had 36 total traps, with 50 m spacing in a N/S direction, and 65 m spacing E/W. We sampled in both habitat and non-habitat.

Our_captures<-sim.capthist(Traps, popn=pop, detectfn=0, detectpar=list(g0=.3, sigma= 25),noccasions=10)

summary(Our_captures)

plot(Our_captures, add=T, tracks=T)

We simulate a capture history, assuming that each of our 36 traps is active for all 10 occasions. We assume that the detection probability of an animal at its activity center is 0.3 per occasion (potentially higher than most real data sets), and that an individual animal has a home range with radius ~60 meters (Thus, there are about 4 traps per home range or detection range for animals living within the array, and fewer for animals along the edges). Unsurprisingly, our detections are generally located where the animals are. However, also note that we detect individuals in non-habitat: while we have allowed density to vary by habitat, our animals still use non-habitat.

fit.1<-secr.fit(Our_captures, model=list(D~1, g0~1, sigma~1), mask=WLE650_Hab, detectfn = 0)

fit.1

fit.2<-secr.fit(Our_captures, model=list(D~1, g0~1, sigma~1), mask=WLE650_Hab, detectfn = 2)

fit.2

We fit models that both assume density is a homogenous point process, but have different detection functions. The first model fits a bivariate normal detection function with relation to distance, and the second fits an exponential function. You can compare or select a detection function based using your standard IT methods, but should not model average any detection parameter across different detection functions because the algebraic formula is different. Model averaging density is fine.

Both models that assume the detection process is solely limited to Euclidean distances, but there are other options: one interesting way to model the detection process is to conceptualize distance as a function of landscape resistance or movement 'cost' (Royle et al. 2013); i.e., animals may be less likely to cross roads or large clearings, or something. The 'secr' vignette on non-euclidean distance functions covers this topic. Similarly, some organisms may use 1-dimensional features like streams; the separate package 'secrlinear' is designed to deal with these unique considerations.

But simpler models can be used to allow group specific detection parameters, varied behavioral/Markovian responses, time specific factors or trends, and trap covariates (either fixed, or time varying, see below). All covariates are fit with a tilde, i.e. 'g0~t'. Trap covariates may also be used to model detection covariates.

'g': group covariates listed in secr.fit argument 'groups'
't': time-specific detection
'T': linear trend in detection
'b': individual trap-happy or shy response across all trap locations
'B': detection is Markovian across all traps
'bk': behavioral response to specific traps
'Bk': trap-specifc markovian response
'k': site effectiveness changes once any animal caught

‘K’: site effectiveness depends on preceding occasion
‘session’: session specific variation (session as fixed effect)
‘Session’: linear detection trend (on link scale) across sessions
‘h2’: 2-part finite mixture model with 2 latent classes

Individual animal covariates can also be used, but models have to be fit using conditional likelihood. Conditional likelihood can be significantly faster, so if for some reason you are not interested in density variance and have a very large dataset, you might choose a secr.fit with ‘CL=TRUE’. As an aside, you might want to think about which specific parameters these covariates influence: in most cases, it is probably more likely that the encounter rate (g0, etc.) is influenced more than the total space that an individual might use.

region.N(fit.1, region=WLE650_Hab)

region.N(fit.2, region=WLE650_Hab)

We estimate population size across the entire mask (or area of inference). Expected N is basically a summation of predicted density (in this case, intercept only). Realized N is very similar to expected N, but the variance associated with the individuals observed is removed (Efford and Fewster 2013); this is not quite the same as the empirical Bayes BUP used in ‘unmarked’. At any rate, the estimates are very comparable: density is generally robust to the detection function, but the derived estimates of individual space use are not. In other words: the detection function doesn’t matter too much when trying to get a sense of how many individual are out there, but a function-specific estimate of individual space use should not be used to infer home range size.

fit.3<-secr.fit(Our_captures, model=list(D~habitat, g0~1, sigma~1), mask=WLE650_Hab, detectfn = 0)

fit.3

Here, we model density as varying in a non-homogenous fashion: basically, habitat and non-habitat have different underlying densities averages and variances. Density models can incorporate any covariates associated with the mask (e.g., ‘D~[covariate1]+[covariate2]+[covariate1]*[covariate2]); the default link function is the log-link. Another alternative is the identity link (where expected densities < 0 are truncated to 0) or splines; different functions can be user-defined or achieved using transformations (see ‘secr’ vignette on density-surfaces).

region.N(fit.3, region=WLE650_Hab)

Note that this abundance estimate is closest to the truth. By fitting a null density model, we predict that there are more undetectable animals out there than there really are.

fit.4<-secr.fit(Our_captures, model=list(D~habitat, g0~1, sigma~1), mask=WLE650_Hab, detectfn = 0, link=list(D="identity")

Fitting the same model as above, but with a linear density function rather than log-linear. This makes little difference for a factorial model, but could be more important if density predictors are continuous.

temp<-secrlist(HN=fit.1, Exp=fit.2, Loglin=fit.3, Lin=fit.4)

AIC(temp)

model.average(temp, betanames="D")

model.average(temp, realnames='D')

We perform some model selection, and note little difference in support between our top models. Note that model averaging produces results that are, ah, *interesting.* We've model averaged across different link functions, which produces some wildly variable predictions given that the formulas used are different. Don't do this in an actual analysis: instead, compare link functions using AIC in the same way you might compare mixing distributions in unmarked.

plot(predictDsurface(fit.3))

This is the average intensity of the spatial point process as predicted by one highly supported model. Not very exciting for a factorial fit.

secr.test(fit.3, nsim=200, fit=F)

###could run the following: secr.test(fit.3, nsim=25, fit=T)

Here, we test goodness-of-fit using only the number of animals captured once (basically, a g-o-f for detection). If fit=T, then we are both simulating and fitting capture histories: much slower, but the default statistic returned is c-hat, which may be useful for adjusting the variance of your estimates and assessing the overall value of your model set.

bootD<-function(object){unlist(predict(object)["D", -1])}

simBoot<-sim.secr(fit.3, nsim=15, extractfn=bootD)

simBoot

quantile(simBoot[,1], c(.025, .975))

This is code to generate bootstrap CI for density. Unfortunately, there's not an option to bootstrap realized N, but this is a way to get a sense of the profile-based density CI when we might have multiple covariates in lieu of the delta method.

fxi.contour(fit.emu3, i=1:15, add=T)

pdot.contour(Traps, detectfn=0, detectpar=list(g0=0.22, sigma=26.8), noccasions=10, levels=seq(0.025, .975, .19), add=T)

The above plots are really window-dressing, but allow visualization of estimated activity center locations and a sense of the effective sampling area. Note that our effective sampling area actually extends slightly beyond our mask: this is likely one reason our point estimates of density and abundance were biased slightly high.

Assignment: There is no assignment for this lab.

REFERENCES

Borchers, D.L. & Efford, M.G. (2008) Spatially-explicit maximum likelihood methods for capture-recapture studies. *Biometrics,* 64, 377-385.

Dorazio, R. M., & J. A. Royle. (2003). Mixture models for estimating the size of a closed population when capture rates vary between individuals. *Biometrics*, 61, 1093-1101.

Efford, M.G. & Fewster, R.M. (2013) Estimating population size by spatially explicit capture-recapture. *Oikos,* 122, 918-928.

Gardner, B., J. A. Royle, and M. T. Wegan. (2009). Hierarchical models for estimating density from DNA mark-recapture studies. *Ecology,* 90, 1106-1115.

Huggins, R. M. (1991). Some practical aspects of a conditional likelihood approach to capture experiments. *Biometrics*, 47, 725-732.

Otis, D. L., Burnham, K.P., White, G.C. & Anderson, D.R. (1978) Statistical inference from capture data on closed animal populations. *Wildlife Monographs*, 62, 3-135.

Pledger, S. (2000). Unified maximum likelihood estimates for closed capture-recapture models using mixtures. *Biometrics,* 56, 434-442.

Link, W. A. (2003). Nonidentifiability of population size from capture-recapture data with heterogeneous detection probabilities. *Biometrics*, 59, 1123-1130.

Royle, J. A., and K. V. Young. (2008). A hierarchical model for spatial capture-recapture data. Ecology 89, 2281-2289.

Royle, J.A., Chandler, R.B., Gazenski, K.D. & Graves, T.A. (2013*a*) Spatial capture-recapture models for jointly estimating population density and landscape connectivity. *Ecology,* 94, 287-294.

Royle, J.A., Chandler, R.B., Sun, C.C. & Fuller, A.K. (2013*b*) Integrating resource selection information with spatial capture-recapture. *Methods in Ecology and Evolution,* 4, 520-530.