A tutorial on fitting spatially explicit capture–recapture models in **secr**

Murray Efford

2021-05-19

Contents

Introduction

This vignette is a guide for those taking their first steps in fitting spatially explicit capture–recapture (SECR) models with the R package **secr** 4.4. The Alaskan snowshoe hare data of Burnham and Cushwa are used as an example. This dataset was first presented by Otis et al. (1978) and has been much used in the exploration of models for heterogeneous capture probability.

SECR data and models

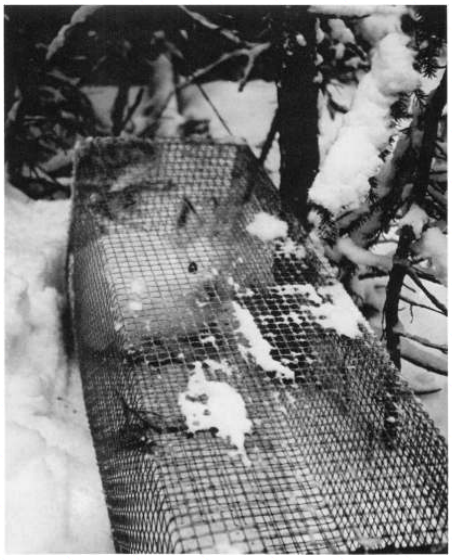
SECR data are observations of marked animals at known locations. Observations result from a well-defined regime of spatial sampling. Sampling is most commonly done with traps, cameras, or some other type of passive detector.

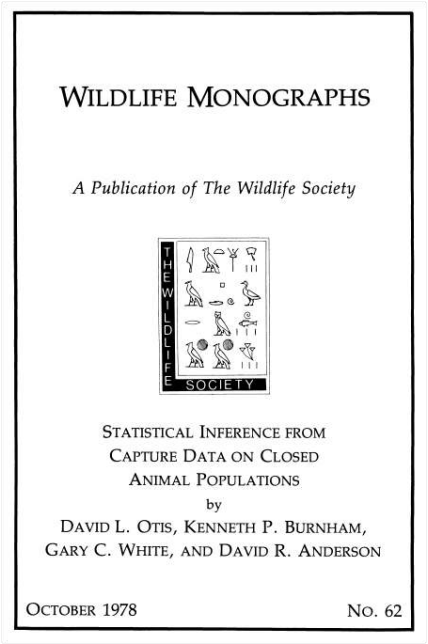
The purpose of the analysis is to estimate parameters of the animal population, particularly the population density. Density is defined as the intensity of a spatial point pattern. Each point represents the enduring location of an animal, its activity center, roughly speaking. In order to estimate density from a sample we must account for the sampling process. The process is inherently spatial: each animal is more likely to be detected near its activity centre, and less likely to be detected far away.

A SECR model combines a model for the point process (the state model) and a model for distance-dependent detection (the observation model). We obtain an unbiased estimate of population density (and other parameters) by jointly fitting the state and observation models.

Although these ideas extend to sampling of open populations, in this document (and throughout the **secr** package) we are concerned only with closed populations. A closed population is one in which the composition of the population and the activity distributions of individuals can be assumed fixed for the duration of sampling.

Snowshoe hare data





> "In 1972, Burnham and Cushwa (pers. comm.) laid out a livetrapping grid in a black spruce forest 30 miles (48.3 km) north of Fairbanks, Alaska. The basic grid was 10 x 10, with traps spaced 200 feet (61 m) apart. Trapping for snowshoe hares *Lepus americanus* was carried out for 9 consecutive days in early winter. Traps were not baited for the first 3 days, and therefore we have chosen to analyze the data from the last 6 days of trapping." Otis et al. (1978:36)

Getting the data together

The raw data have been prepared as two text files[1]. They may be downloaded from https://www.otago.ac.nz/density/examples/. We assume the files are in your R working directory; use `setwd` to change that if necessary.

[1]These data were provided with the CAPTURE software and have been reshaped into the standard input format for DENSITY (Efford 2012) and **secr** using the code in Appendix 1.

The capture file "hareCH6capt.txt" has one line per capture and four columns (the header lines are commented out and are not needed). Here we display the first 6 lines. The first column is a session label derived from the original study name; it becomes significant for multi-session datasets (secr-multisession.pdf).

```
# Burnham and Cushwa snowshoe hare captures
# Session ID Occasion Detector
wickershamunburne  1 2 0201
wickershamunburne 19 1 0501
wickershamunburne 72 5 0601
wickershamunburne 73 6 0601
...
```

The trap layout file "hareCH6trap.txt" has one row per trap and columns for the detector label and x- and y-coordinates. We display the first 6 lines. The detector label is used to link captures to trap sites. Coordinates can relate to any rectangular coordinate system; **secr** will assume distances are in metres. These coordinates simply describe a 10×10 square grid with spacing 60.96 m. Do not use unprojected geographic coordinates (latitude and longitude)[2].

```
# Burnham and Cushwa snowshoe hare trap layout
# Detector  x  y
0101 0 0
0201 60.96 0
0301 121.92 0
0401 182.88 0
...
```

We can now load **secr** and read the data files to construct a capthist object. The detectors are single-catch traps (maximum of one capture per animal per occasion and one capture per trap per occasion).

```
library(secr)
```

```
## This is secr 4.4.3. For overview type ?secr
```

```
hareCH6 <- read.capthist("hareCH6capt.txt", "hareCH6trap.txt", detector = "single")
```

```
## No errors found :-)
```

The capthist object `hareCH6` now contains almost all the information we need to fit a model. However, before launching into that it's good to take a deep breath and examine the raw data.

Exploring the data

The data should first be summarised and plotted.

```
summary(hareCH6)
```

```
## Object class       capthist
## Detector type      single
## Detector number    100
## Average spacing    60.96 m
## x-range            0 548.64 m
## y-range            0 548.64 m
##
## Counts by occasion
##                      1   2   3   4   5   6 Total
```

[2]Function `spTransform` in package **rgdal** may help.

```
## n                   16  28  20  26  23  32   145
## u                   16  24   9   9   6   4    68
## f                   25  22  13   5   1   2    68
## M(t+1)              16  40  49  58  64  68    68
## losses               0   0   0   0   0   0     0
## detections          16  28  20  26  23  32   145
## detectors visited   16  28  20  26  23  32   145
## detectors used     100 100 100 100 100 100   600
```

The last column ('Total') is a simple sum over other columns except for M(t+1).
The counts 'n', 'u', 'f' and 'M(t+1)' will make perfect sense if you are familiar with Otis et al. (1978), but just in case you're not...

Table 1. Summary counts

Count	Description
n	number of distinct individuals detected on each occasion t
u	number of individuals detected for the first time on each occasion t
f	number of individuals detected on exactly t occasions
M(t+1)	cumulative number of detected individuals on each occasion t

The conventional summary counts are all well and good[3], but these are *spatial* data so we learn a lot by mapping them. We use the `plot` method, which for capthist objects has additional arguments; we set `tracks = TRUE` to join consecutive captures of each individual.

```
par(mar = c(1,1,3,1))  # reduce margins
plot (hareCH6, tracks = TRUE)
```

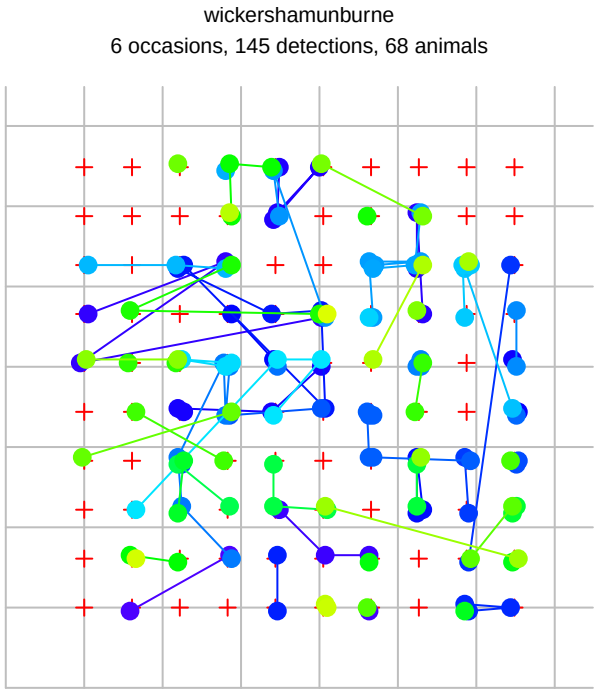


Fig. 1. Snowshoe hare spatial capture data plotted in **secr**. Trap sites (red crosses) are 61 m apart. Grid

[3]And the minimum sufficient statistics for the non-spatial estimators in Otis et al. (1978) are functions of these counts.

lines (grey) are 100 m apart (use arguments `gridl` and `gridsp` to suppress the grid or vary its spacing). Colours help distinguish individuals, but some are recycled.

The most important insight from Fig. 1 is that individuals tend to be recaptured near their site of first capture. This is expected when the individuals of a species occupy home ranges. In SECR models the tendency for detections to be localised is reflected in the spatial scale parameter σ. Good estimation of σ and density D requires spatial recaptures (i.e. captures at sites other than the site of first capture).

Successive trap-revealed movements can be extracted with the `moves` function and summarised with `hist`:

```
m <- unlist(moves(hareCH6))
par(mar = c(3.2,4,1,1), mgp = c(2.1,0.6,0))  # reduce margins
hist(m, breaks = seq(-61/2, 500,61), xlab = "Movement  m", main = "")
```

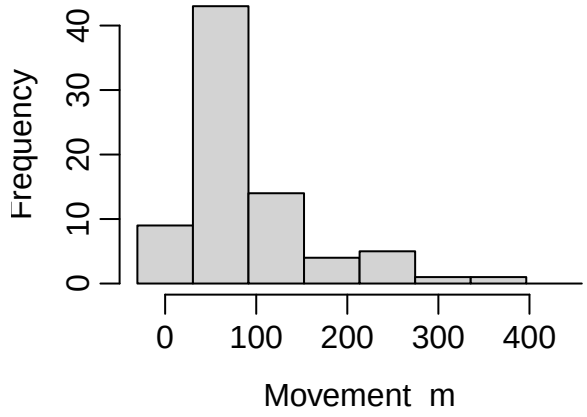


Fig. 2. Successive trap-revealed movements of snowshoe hares on 61-m grid.

About 30% of trap-revealed movements were of > 100 m (Fig. 2; try also `plot(ecdf(m))`), so we can be sure that peripheral hares stood a good chance of being trapped even if their home ranges were centred well outside the area plotted in Fig. 1.

The function `RPSV` with option CC = TRUE provides a biased estimate of the spatial scale σ, ignoring the problem that movements are truncated by the edge of the grid:

```
initialsigma <- RPSV(hareCH6, CC = TRUE)
cat("Quick and biased estimate of sigma =", initialsigma, "m\n")
```

```
## Quick and biased estimate of sigma = 63.6612 m
```

This estimate will be useful when we come to fit a model.

Fitting a simple model

Calling `secr.fit`

Next we fit the simplest possible SECR model with function `secr.fit`. Setting `trace = FALSE` suppresses printing of intermediate likelihood evaluations; it doesn't hurt to leave it out. We save the fitted model with the name 'fit'.

```
fit <- secr.fit (hareCH6, buffer = 4 * initialsigma, trace = FALSE)
```

```
## Warning in secr.fit(hareCH6, buffer = 4 * initialsigma, trace = FALSE): multi-catch
## likelihood used for single-catch traps
```

A warning is generated. The data are from single-catch traps, but there is no usable theory for likelihood-based estimation from single-catch traps. This is not the obstacle it might seem, because simulations seem to show

that the alternative likelihood for multi-catch traps may be used without damaging the density estimates (Efford, Borchers and Byrom 2009). It is safe to ignore the warning for now[4]. In order to avoid the warning in later fits we reset the detector type to "multi".

```
detector(traps(hareCH6)) <- "multi"
```

Reviewing the output

The output from `secr.fit` is an object of class 'secr' (confirm this with `class(fit)`). If you investigate the structure of `fit` with `str(fit)` it will seem to be a mess: it is a list with more than 25 components, none of which contains the final estimates you are looking for.

To examine model output or extract particular results you should use one of the functions defined for the purpose. Technically, these are S3 methods for the class 'secr'. The key methods are `print`, `plot`, `AIC`, `coef`, `vcov` and `predict`. Append '.secr' when seeking help e.g. `?print.secr`.

Typing the name of the fitted model at the R prompt invokes the print method for `secr` objects and displays a more useful report.

```
fit
```

```
##
## secr.fit(capthist = hareCH6, buffer = 4 * initialsigma, trace = FALSE)
## secr 4.4.3, 17:18:55 19 May 2021
##
## Detector type      single
## Detector number    100
## Average spacing    60.96 m
## x-range            0 548.64 m
## y-range            0 548.64 m
##
## N animals       :  68
## N detections    :  145
## N occasions     :  6
## Mask area       :  106.129 ha
##
## Model           :  D~1 g0~1 sigma~1
## Fixed (real)    :  none
## Detection fn    :  halfnormal
## Distribution    :  poisson
## N parameters    :  3
## Log likelihood  :  -607.988
## AIC             :  1221.98
## AICc            :  1222.35
##
## Beta parameters (coefficients)
##            beta   SE.beta       lcl       ucl
## D      0.382511 0.1299543  0.127805  0.637217
## g0    -2.723724 0.1609129 -3.039107 -2.408340
## sigma  4.224543 0.0653075  4.096542  4.352543
##
## Variance-covariance matrix of beta parameters
##                  D          g0       sigma
## D      0.01688813 -0.00172941 -0.00162420
## g0    -0.00172941  0.02589295 -0.00737210
```

[4]While noting that estimates of the detection parameter g0 are biased.

```
## sigma -0.00162420 -0.00737210  0.00426506
##
## Fitted (real) parameters evaluated at base levels of covariates
##        link   estimate SE.estimate        lcl        ucl
## D       log  1.4659614  0.19131520  1.1363320  1.8912105
## g0    logit  0.0615879  0.00929993  0.0456901  0.0825389
## sigma   log 68.3432307  4.46808544 60.1320074 77.6757235
```

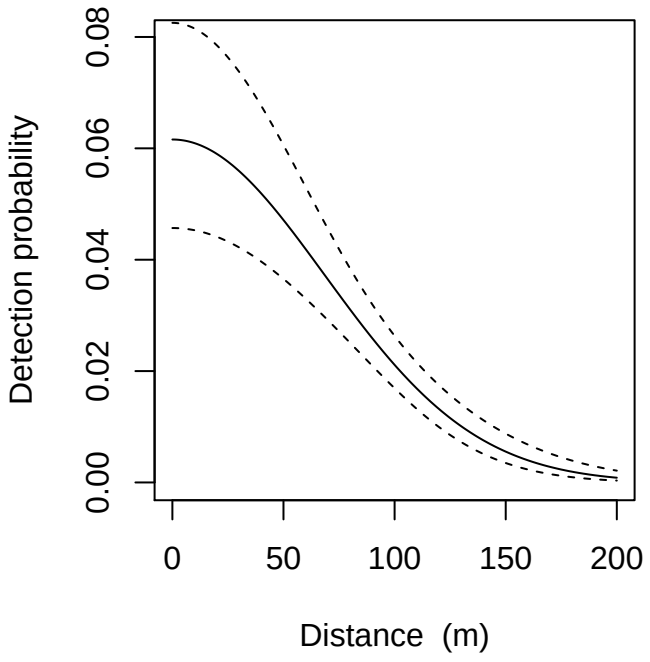
The report comprises these sections that you should identify:

- function call and time stamp
- summary of the data
- description of the model, including the maximized log likelihood, Akaike's Information Criterion AIC
- estimates of model coefficients (beta parameters)
- estimates of variance-covariance matrix of the coefficients
- estimates of the 'real' parameters

The last three items are generated by the `coef`, `vcov` and `predict` methods respectively. The final table of estimates is the most interesting, but it is derived from the other two. For our simple model there is one beta parameter for each real parameter[5]. The estimated density is 1.47 hares per hectare, 95% confidence interval 1.14–1.89 hares per hectare[6].

The other two real parameters jointly determine the detection function that you can easily plot with 95% confidence limits:

```
par(mar = c(4,4,1,1))  # reduce margins
plot(fit, limits = TRUE)
```



[5]We can get from beta parameter estimates to real parameter estimates by applying the inverse of the link function e.g. $\hat{D} = \exp(\hat{\beta}_D)$, and similarly for confidence limits; standard errors require a delta-method approximation (Lebreton et al. 1992).

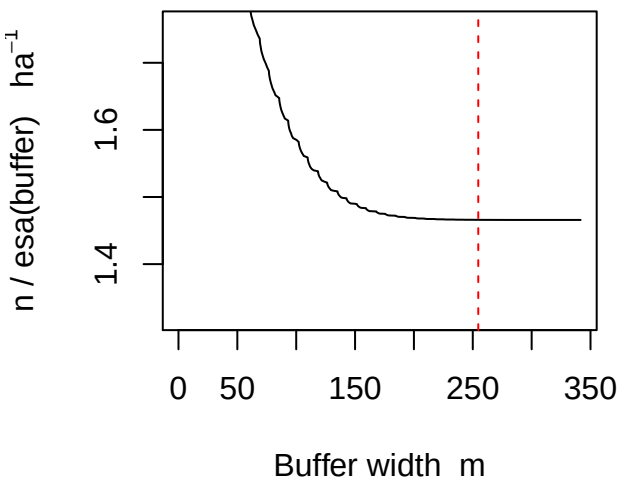
[6]One hectare (ha) is 10000 m^2 or 0.01 km^2.

Choosing the buffer width

When fitting the simple model with `secr.fit` we used `buffer = 4 * initialsigma` without giving the explanation. Here it is. As far as we know, the snowshow hare traps were surrounded by suitable habitat. We limit our attention to the area immediately around the traps by specifying a habitat buffer. The `buffer` argument is a short-cut method for defining the area of integration; the alternative is to provide a habitat mask in the `mask` argument. Buffers and habitat masks are covered at length in secr-habitatmasks.pdf.

The theory of SECR tells us that buffer width is not critical as long as it is wide enough that animals at the edge have effectively zero chance of appearing in our sample. The 4σ suggestion is based on experience with half-normal detection models[7]. We check that for the present model with the function `esa.plot`. The estimated density[8] has easily reached a plateau at the chosen buffer width (dashed red line):

```
esa.plot(fit)
abline(v = 4 * initialsigma, lty = 2, col = 'red')
```



Choosing a detection function

- detection probability declines with distance according to a half-normal curve

We can try alternative shapes for the detection function (the decline in detection probability with distance).

secr offers several different shapes of detection function (see the list at `?detectfn`). We need to sort these out. All except ANN and HAN decline monotonically with distance. Three are only used for acoustic data (BSS, SS, SSS). The simplest UN is not available for maximum likelihood model fitting, and several are frankly exotic and almost never used (CHN, WEX, CLN, CG), as are ANN and HAN.

That leaves the half-normal, negative exponential, and hazard rate functions (HN, EX, HR) These differ primarily in the length of their tails i.e. the probability they assign to very distant detections. The half-normal makes distant detections very improbable, the negative exponential less so; The 'hazard-rate' function requires a third parameter and potentially has a very long tail indeed.

Fit each of these and assess the effect. We use a wider buffer to allow for longer tails.

```
fit.HN <- secr.fit (hareCH6, buffer = 6 * initialsigma, detectfn = 'HN', trace = FALSE)
fit.EX <- secr.fit (hareCH6, buffer = 6 * initialsigma, detectfn = 'EX', trace = FALSE)
fit.HR <- secr.fit (hareCH6, buffer = 6 * initialsigma, detectfn = 'HR', trace = FALSE)
```

[7] This is not just the tail probability of a normal deviate; think about how the probability of an individual being detected at least once changes with (i) the duration of sampling (ii) the density of detector array.

[8] These are Horvitz-Thompson-like estimates of density obtained by dividing the observed number of individuals n by effective sampling areas (Borchers and Efford 2008) computed as the cumulative sum over mask cells ordered by distance from the traps. The algorithm treats the detection parameters as known and fixed.

How do the models compare? The last one (fit.HR) raised a warning from the post-fitting bias check that we discuss more below. We bundle the fits together in an object of class secrlist – this is simply a convenience – and then inspect the estimates:

```
fits <- secrlist(HN = fit.HN, EX = fit.EX, HR = fit.HR)
predict(fits)
```

```
## $HN
##        link   estimate SE.estimate        lcl       ucl
## D       log  1.4659033  0.19132892  1.1362550  1.891189
## g0    logit  0.0616056  0.00929655  0.0457122  0.082547
## sigma   log 68.3310481  4.46116307 60.1318198 77.648276
##
## $EX
##        link  estimate SE.estimate       lcl       ucl
## D       log  1.477147   0.1949283  1.141777  1.911024
## g0    logit  0.179717   0.0309931  0.126715  0.248579
## sigma   log 39.917894   3.2925569 33.968529 46.909251
##
## $HR
##        link  estimate SE.estimate        lcl      ucl
## D       log  1.354909   0.1936607  1.0253223  1.79044
## g0    logit  0.129105   0.0364613  0.0727981  0.21869
## sigma   log 45.366628   8.4352435 31.6090342 65.11211
## z       log  3.115165   0.3134248  2.5589101  3.79234
```

Note how similar the density estimates are from HN and EX; HR not so much. The parameter named 'sigma' means a different thing for each function, so do not compare. To my mind this also applies to the function-specific 'g0'.

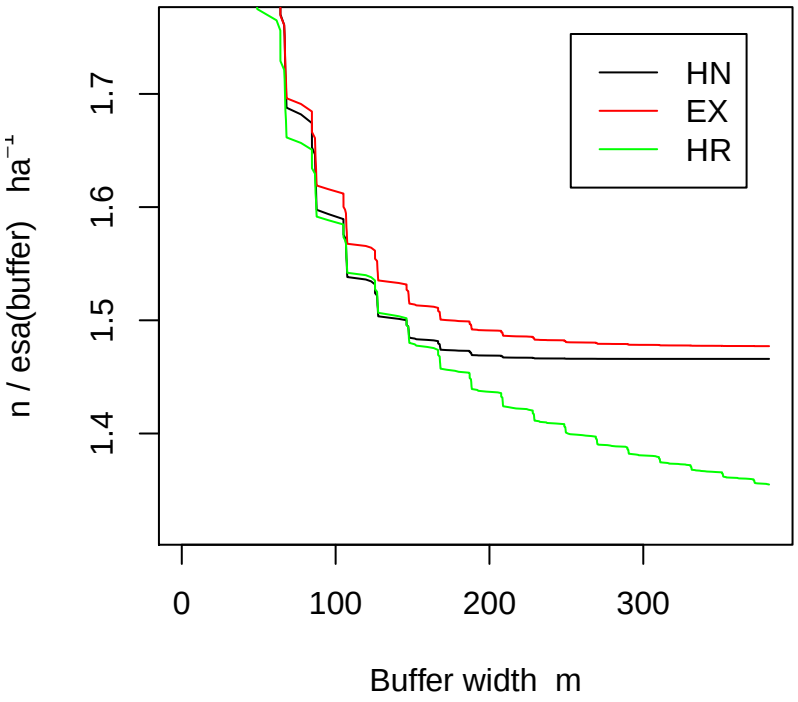
Formal model comparison by AIC places the longer-tailed functions EX and HR ahead of HN:

```
AIC(fits)
```

```
##                   model    detectfn npar   logLik     AIC    AICc  dAICc AICcwt
## HR D~1 g0~1 sigma~1 z~1 hazard rate    4 -599.880 1207.76 1208.39  0.000 0.6467
## EX     D~1 g0~1 sigma~1 exponential    3 -601.615 1209.23 1209.60  1.209 0.3533
## HN     D~1 g0~1 sigma~1  halfnormal    3 -607.991 1221.98 1222.36 13.963 0.0000
```

The selection of HR is a worry because it has some annoying properties as we can see on this plot:

```
par(mar = c(4,4,2,2))
esa.plot(fits, max.buffer = 6 * initialsigma)
```



Although the density estimates from HN and EX reach a plateau fairly promptly with increasing buffer width, the estimates from HR do not. That means that estimates of density remain sensitive to buffer width out to quite large distances. The sensitivity explains why the bias check raised a warning. It is an argument for not using HR except where there is a natural boundary (check out habitat islands in secr-habitatmasks.pdf). There is an unresolved research question here: Do real animals have such long tails?

Provisionally rejecting HR, we are left with EX as the preferred model for these data. HN delivers essentially the same estimate of density.

The `model` argument of `secr.fit`

Our initial model assumed that the detection of all individuals is governed by the same detection vs distance curve at all detectors on all occasions. The 'model' argument of `secr.fit` allows this assumption to be relaxed in particular ways.

Although this is a longish section, and one that can occupy a lot of time, be warned that the returns from exhaustively pursuing the last sliver of improvement in fit are usually trivial. We rely on the robustness of SECR models. The potential effects of most importance are learned responses and individual heterogeneity.

The 'model' argument allows us to specify variation in each of the 'real' parameters (i.e. density D and the detection parameters g0 [9] and sigma). In the default model each real parameter is constant (D ~ 1, g0 ~ 1, sigma ~ 1), but the constant indicator '1' in each formula may be replaced by a predictor, or perhaps a combination of predictors. We focus on the detection parameters. The predictor may be

- a code for an automatically generated predictor, or
- the name of an individual, trap or session covariate.

Codes for automatically generated predictors are listed on the help page `?"secr detection models"`, and we show a subset in Table 2. The most important in terms of an effect on estimates of density are those for learned responses and unmodelled individual heterogeneity. Heterogeneity raises multiple issues that we do not have space for here. We concentrate on learned responses.

[9] For some detection functions g0 is replaced by lambda0

Table 2. Automatic predictors commonly used to model detection parameters.

Code	Description
b	permanent global learned response
bk	permanent detector-specific learned response
t	time factor (one level for each occasion)
T	time trend (integer covariate 0:(S-1))
g	group (as specified by 'groups' argument)
h2	2-class finite mixture
session	session factor (one level for each session)

Learned responses

Otis et al. (1978) considered the possibility that the experience of capture induced a change in the probability of capturing an individual on any later occasion. They called this a 'behavioral response', and henceforth it has been labelled 'b'. The model envisaged a permanent step change in behaviour

Spatial models may include more subtle effects. Most importantly, the learned response may be specific to the detector location (code bk), rather than applying generally across all detectors (code b).

We consider the possibility of either sort of learned response in snowshoe hares:

```
fit.EXb <- secr.fit (hareCH6, buffer = 6 * initialsigma, detectfn = 'EX',
                     model = g0 ~ b, trace = FALSE)
fit.EXbk <- secr.fit (hareCH6, buffer = 6 * initialsigma, detectfn = 'EX',
                      model = g0 ~ bk, trace = FALSE)
```

For convenience we bundle the original (null) model and the two new models together in one object of class 'secrlist'. That may be passed as a unit to other functions, particularly AIC:

```
fitsb <- secrlist(null = fits$EX, b = fit.EXb, bk = fit.EXbk)
AIC(fitsb)
```

```
##                  model    detectfn npar   logLik     AIC    AICc dAICc AICcwt
## b     D~1 g0~b sigma~1 exponential    4 -599.824 1207.65 1208.28 0.000 0.5004
## null  D~1 g0~1 sigma~1 exponential    3 -601.615 1209.23 1209.60 1.321 0.2585
## bk   D~1 g0~bk sigma~1 exponential    4 -600.554 1209.11 1209.74 1.461 0.2410
```

```
predict(fitsb)
```

```
## $null
##        link  estimate SE.estimate       lcl       ucl
## D       log  1.477147   0.1949283  1.141777  1.911024
## g0    logit  0.179717   0.0309931  0.126715  0.248579
## sigma   log 39.917894   3.2925569 33.968529 46.909251
##
## $b
##        link  estimate SE.estimate        lcl       ucl
## D       log  1.761363   0.3431841  1.2065444  2.571311
## g0    logit  0.117373   0.0393706  0.0593902  0.218798
## sigma   log 40.158835   3.3229382 34.1561292 47.216476
##
## $bk
##        link  estimate SE.estimate        lcl      ucl
## D       log  1.482522   0.1988793  1.1410948  1.92611
```

```
## g0    logit  0.152245   0.0317557  0.0998103  0.22533
## sigma   log 42.368890   4.0096793 35.2104135 50.98272
```

The global response model b comes out on top. However, the AIC differences among the three models (b, bk, null) are very small. This would not be a problem, except that the density estimate from model b is noticeably larger than the others, so it does matter which we choose. The learned response may be positive or negative. The direction can be determined by the sign of the relevant coefficient (g0.bTRUE) in `coef(fitsb$b)`. The coefficient is 0.61 (95% CI -0.1 – 1.32), remembering that this relates to the link (logit) scale.

Results for predictor levels other than the base level

The default output from `predict` for estimates of the 'real' parameters is incomplete: it shows only the value of g0 for a naive animal (b = 0). To see the estimates for both b = 0 and b = 1, and hence the magnitude of the effect on the probability scale, we specify the 'newdata' argument of the `predict` method:[10]

```
predict(fitsb$b, newdata = data.frame(b = 0:1), realnames = "g0")
```

```
## $`b = 0`
##     link estimate SE.estimate       lcl      ucl
## g0 logit 0.117373   0.0393706 0.0593902 0.218798
##
## $`b = 1`
##     link estimate SE.estimate      lcl      ucl
## g0 logit 0.196379   0.0349557 0.136672 0.273895
```

It will usually be necessary to specify 'newdata' like this to obtain the predicted values of real parameters at different levels of the predictors. `expand.grid` is a handy alternative to `data.frame` if you want to see all combinations of predictors.

Model averaging

One way to duck the problem of selecting a single model is to average over the models using AIC model weights. There is a function for this:

```
model.average(fitsb)
```

```
##       estimate SE.estimate        lcl       ucl
## D      1.61447   0.3046817  1.1189105  2.329512
## g0     0.13993   0.0456512  0.0718024  0.254944
## sigma 40.60245   3.6132539 34.1155418 48.322814
```

Our model-informed 'best guess' of the density comes out at 1.61 hares per hectare with 95% confidence interval 1.12–2.33 hares per hectare.

References

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

Efford, M. G. (2012) Efford MG 2012. DENSITY 5.0: software for spatially explicit capture-recapture. Department of Mathematics and Statistics, University of Otago, Dunedin, New Zealand. https://www.otago.ac.nz/density.

Efford, M. G., Borchers D. L. and Byrom, A. E. (2009) Density estimation by spatially explicit capture–recapture: likelihood-based methods. In: D. L. Thomson, E. G. Cooch and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer. Pp. 255–269.

[10] We also use the new argument 'realnames' to select only the parameter we want.

Lebreton, J.-D., Burnham, K. P., Clobert, J. and Anderson, D. R. (1992) Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs* **62**, 67–118.

Otis, D. L., Burnham, K. P., White, G. C. and Anderson, D. R. (1978) Statistical inference from capture data on closed animal populations. *Wildlife Monographs* No. **62**.

Appendix 1. Conversion of snowshoe hare data from CAPTURE format.

First define a function to peek at text files:

```
displayLines <- function(filename, nlines, final = "") {
    con <- file(filename)
    cat(readLines(con,nlines), sep='\n')
    if (final != "") cat(final, "\n")
    close(con)
}
```

```
library(secr)
workdir <- 'd:/density secr 4.4/package data/snowshoehares/'
inpfile <- paste0(workdir, 'hares.txt')
# check raw data
displayLines(inpfile, 9)
```

```
## data='Data from Burnham and Cushwa (in prep.), Lepus americanus, inter. Alaska.'
## format='(33x,a2,5x,18f2.0)'
## read input data
## wickersham unburned fall 1975     1      4 1 0 0 5 2 0 0 2 1 0 0 0 0 4 2 0 0
## wickersham unburned fall 1975     2      3 2 0 0 0 0 0 0 7 1 0 0 0 0 0 0 0 0
## wickersham unburned fall 1975     3      5 2 0 0 0 0 0 0 0 0 0 0 0 7 2 6 2 5 3
## wickersham unburned fall 1975     4      5 8 0 0 0 0 0 0 6 7 1 6 4 8 0 0 1 7
## wickersham unburned fall 1975     5      5 9 0 0 0 0 0 0 8 8 0 0 8 9 0 0 8 7
## wickersham unburned fall 1975     6      6 9 0 0 0 0 0 0 5 9 610 0 0 5 9 510
```

```
# read raw data, skipping 3 header lines, and shape into a dataframe with one row
# per capture
tmp <- read.fortran(inpfile, format = c('A29','4X', 'A2','5X','9A4'), skip = 3)
capt <- data.frame(session = rep(tmp[,1], each = 9),
                   ID = rep(tmp[,2], each=9),
                   occasion = rep(1:9, nrow(tmp)),
                   trapID = as.character(t(tmp[,3:11])),
                   stringsAsFactors = FALSE)
# drop non-captures
capt <- capt[capt$trapID!= ' 0 0',]
# recode blanks to zero in trapID,
capt$trapID <- gsub(' ','0', capt$trapID)
# view first 6 captures (ordered by animal)
head(capt)
```

```
##                          session ID occasion trapID
## 1  wickersham unburned fall 1975  1        1   0401
## 3  wickersham unburned fall 1975  1        3   0502
## 5  wickersham unburned fall 1975  1        5   0201
## 8  wickersham unburned fall 1975  1        8   0402
## 10 wickersham unburned fall 1975  2        1   0302
## 14 wickersham unburned fall 1975  2        5   0701
```

```
# make trapping grid object
grid <- make.grid(nx = 10, ny = 10, spacing = 60.96, detector = 'single', ID = 'xy')
# construct capthist object
hareCH <- make.capthist(capt, grid)
# restrict to last 6 days
```

```
hareCH6 <- subset(hareCH, occasions = 4:9)
write.capthist(hareCH6)
```

Appendix 2. List of key functions used in this tutorial.

Function	Purpose
`AIC`*	model selection, model weights
`derived`*	compute density from effective sampling area
`esa.plot`	cumulative plot esa or $\hat{D}$ vs buffer width
`model.average`	combine estimates using AIC or AICc weights
`moves`	distances between capture locations
`plot`*	plot 'capthist', 'traps' or 'mask'
`predict`*	'real' parameters for arbitrary levels of predictor variables
`print`*	
`read.capthist`	build capthist object from text files
`RPSV`	'root pooled spatial variance', with CC = TRUE an estimate of $\sigma_H N$
`secr.fit`	maximum likelihood fit; result is a fitted 'secr' object
`summary`*	summarise 'capthist', 'traps' or 'mask'

* S3 method