# **secr** - spatially explicit capture–recapture in R

Murray Efford

January 21, 2013

This document provides an overview of **secr** 2.5, an R package for spatially explicit capture–recapture analysis (SECR). It includes some background on SECR, an outline of the package, and a more detailed description of how models are implemented. See Appendix 1 for a glimpse of **secr** in action. For details of how to use **secr** see the help pages and vignettes.

## Contents

# Introduction to SECR

Spatially explicit capture–recapture (SECR) is a set of methods for modelling animal capture–recapture data collected with an array of 'detectors'. Their main use is in estimating population density, but they also have advantages over non-spatial methods when the goal is to estimate population size (Efford and Fewster 2012). SECR methods overcome edge effects that are problematic in conventional capture–recapture estimation of animal populations (Otis et al. 1978). Detectors may be live-capture traps, with animals uniquely tagged, sticky traps or snags that passively sample hair, from which individuals are distinguished by their microsatellite DNA, or cameras that take photographs from which individuals are recognized by their natural marks. The concept of a detector extends to polygons or transects that are searched for animals or their sign.

The primary data for SECR are (i) the locations of the detectors, and (ii) detections of known individuals on one or more sampling occasions (i.e. their detection histories). The generic terms 'detector' and 'detections' cover several possibilities (see 'Detector types' below); we use them interchangeably with the more specific and familiar terms 'traps' and 'captures'. Table 1 gives a concrete example of trapping data (the structure differs for detectors that are not traps).

In SECR, a spatial model of the population and a spatial model of the detection process are fitted to the spatial detection histories. The resulting estimates of population density are unbiased by edge effects and incomplete detection (other sources of bias may remain). Inverse prediction (IP SECR) and maximum likelihood (ML SECR) are alternative methods for fitting the

Table 1: Example of spatially explicit detection data. Each entry (e.g. A9) records the detector at which a known animal (ID) was observed at each sample time (occasion). '.' indicates no detection. Each detector has known x-y coordinates. Formats for data input are described in 'secr-datainput.pdf'

| | Occasions | | | | |
|---|---|---|---|---|---|
| ID | 1 | 2 | 3 | 4 | 5 |
| 1 | A9 | . | . | . | . |
| 2 | A12 | A12 | . | . | . |
| 3 | . | . | C6 | B5 | . |
| 4 | . | . | G3 | . | F3 |
| etc. | | | | | |

spatial detection model (Efford 2004, Borchers and Efford 2008). Of these, ML SECR is the more flexible, with a caveat for data from single-catch traps. Data augmentation and Markov chain Monte Carlo (MCMC) methods have also been used for SECR (Royle and Young 2008, Royle et al. 2009, Singh et al. 2010), but this approach is orders of magnitude slower than ML SECR and easy to misuse; it is not considered here.

## State and observation models

Like other statistical methods for estimating animal abundance (Borchers et al. 2002), SECR combines a state model and an observation model. The state model describes the distribution of animal home ranges in the landscape, and the observation model (a spatial detection model) relates the probability of detecting an individual at a particular detector to the distance of the detector from a central point in each animal's home range. The distances are not observed directly (usually we don't know the range centres), so conventional distance sampling methods do not apply.

## Distribution of home-range centres

The distribution of range centres in the population (Borchers and Efford 2008) will usually be treated as a homogeneous Poisson point process (Fig. 1). Density is the sole parameter of a Poisson process. An inhomogeneous distribution may also be fitted; this provides a means to evaluate the effects of habitat variables on density.

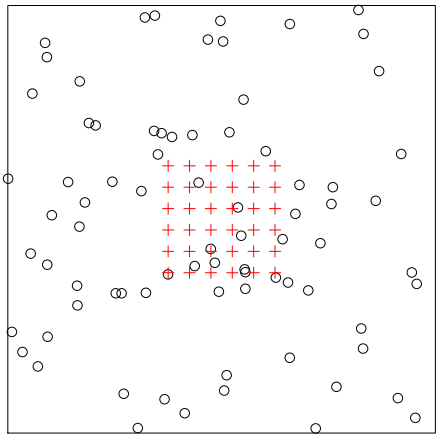

Figure 1: Hypothetical Poisson distribution of range centres near an array of detectors. We estimate the intensity (density) of this distribution.

## Detection functions

A detection model uses one of several possible parametric forms for the decline in detection probability with distance ($d$) from the home-range centre (Table 2, Fig. 2). The probability $g(d)$ is for the 'ideal' case of just one animal and one detector; the actual probability may differ (see discussion of 'traps' under Detector Types).

Table 2: Two functions relating the probability of detection to distance ($d$). See `?detectfn` for more.

| | |
|---|---|
| Halfnormal | $g(d) = g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right)$ |
| Exponential | $g(d) = g_0 \exp\left(-\frac{d}{\sigma}\right)$ |

## Detector types

The properties of detectors are an important part of the SECR observation model. Inside **secr**, data are tagged with a detector type to ensure they are printed, plotted and analysed appropriately.

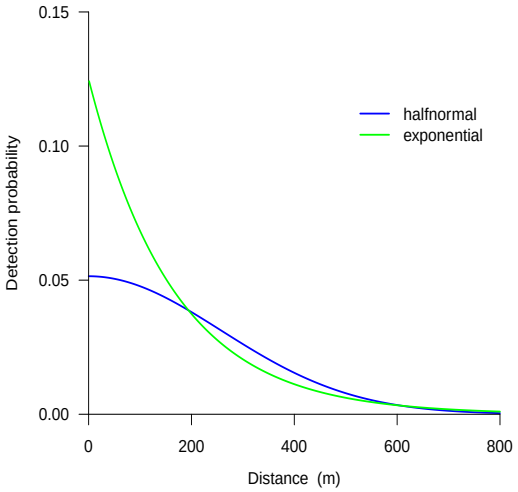


Figure 2: Alternative shapes for a function relating the probability of detection to distance from range centre.

Some common detectors (camera 'traps' and hair snags for DNA) do not capture animals, but merely record when they pass by. These 'proximity' detectors can be considered to act independently of each other. With proximity detectors, each animal × occasion 'cell' of a detection history potentially contains several positive records. In the simplest case each cell contains a binary vector coding presence or absence at each detector. A 'count' detector is a generalised proximity detector in which the data are vectors of counts, one per detector. Models for 'count' data will specify a distribution for the counts (the 'binomN' argument of secr.fit, where binomN=0 indicates Poisson, binomN=1 Bernoulli etc.).

Detectors that are true traps do not act independently because capture of an animal in one trap prevents it being caught in another trap until it is released. Traps expose animals to competing risks of capture. The per-trap probability of capture may be adjusted for the competing risk from other traps by using an additive hazard model (Borchers and Efford 2008). However, if the detectors are traps that catch only one animal at a time then there is a further level of competition – between animals for traps. Multi-catch and single-catch traps therefore represent distinct detector types. No general adjustment has been found for the per-trap probability of capture in the single-catch case (it's an open research question), and there is strictly no known maximum likelihood estimator. However, density estimates using the multi-catch likelihood for single-catch data appear only slightly biased (Efford, Borchers and Byrom 2009).

Polygon and transect detectors are for binary or count detection data (e.g.,

number of detections per animal per polygon per occasion) supplemented with the x-y coordinates of each detection (in the case of a transect it is enough to record the distance along the line). When a study uses multiple search areas or multiple transects, detections may be either independent or dependent (e.g., maximum one per animal per polygon per occasion) as with traps. The dependent or 'exclusive' type is indicated by the suffix 'X'; in this case the counts are necessarily binary. Using the 'polygonX' or 'transectX' detector type ensures that a competing-risk model is fitted.

Acoustic 'signal strength' detectors produce a binary detection vector supplemented by measurements of signal strength, as from an array of microphones.

There is limited support for 'unmarked', 'presence' and 'telemetry' detector types, but these are not yet fully documented. The 'telemetry' detector type is like a 'polygon' detector (detections have x-y coordinates); perimeter coordinates are required, but they are not at present used in analyses. Telemetry data are used to augment capture–recapture data (see `addTelemetry`).

Table 3: Detector types

| | |
|---|---|
| single | traps that catch one animal at a time |
| multi | traps that may catch more than one animal at a time |
| proximity | records presence at a point without restricting movement |
| count | proximity detector allowing >1 detection per animal per time |
| polygon | counts from searching one or more areas |
| transect | counts from searching one or more transects |
| polygonX | binary data from mutually exclusive areas |
| transectX | binary data from mutually exclusive transects |
| signal | detections and signal strengths at multiple microphones |
| telemetry | locations from radiotelemetry |

## Origins and outline of the package 'secr'

The program DENSITY (Efford et al. 2004, Efford 2012) provides a graphical interface to SECR methods that has been accepted by many biologists. However, DENSITY has significant drawbacks: it requires the Windows operating system, its algorithms are not always transparent or well-documented, it fits only homogeneous Poisson models, and it omits some recent advances in SECR.

The R package **secr** was written to address these weaknesses and allow for further development. It implements almost all the methods described by Borchers and Efford (2008), Efford et al. (2009), and Efford (2011). **secr** uses

external C code for computationally intensive operations. Appendix 2 compares the features of DENSITY and **secr**. The important functions of **secr** are listed in Appendix 3.

## How secr works

**secr** defines a set of R classes[1] and methods for data from detector arrays. The essential classes are:

| | |
|---|---|
| `traps` | locations of detectors; detector type ('proximity', 'multi', etc.) |
| `capthist` | spatial detection histories, including a `traps` object |
| `mask` | points on habitat mask |
| `secr` | fitted SECR model. |

To perform an SECR analysis you explicitly or implicitly construct each of these objects in turn, using the functions provided (e.g., `read.capthist`[2], `secr.fit`). Fig. 3 summarizes the relationships among the core object classes. The classes `traps` (not shown), `capthist` and `mask` may optionally store covariates specific to detectors, animals and habitat points respectively. Each set of covariates is saved in a dataframe that is an attribute of the corresponding object; the `covariates` method is used to extract or replace covariates.

## Input

Data input is covered in the separate document 'secr-datainput.pdf'. One option is to use text files in the formats used by DENSITY; these accommodate most types of data. Two files are required, one of detector (trap) locations and one of the detections (captures) themselves; the function `read.capthist` reads both files and constructs a capthist object. Previously (before **secr** 1.4) it was necessary to construct the capthist object in two stages, first making a traps object (with `read.traps`) and a captures dataframe, and then combining these with `make.capthist`. This route is still available for tricky datasets.

[1] Technically, these are S3 classes. A 'class' specifies a particular type of data object and the functions (methods) by which it is manipulated (computed, printed, plotted etc). See the R documentation for further explanation

[2] Text in `teletype` font refers to R objects that are documented in online help for the **secr** package, or in base R. A good place to start is the page for `secr.fit`

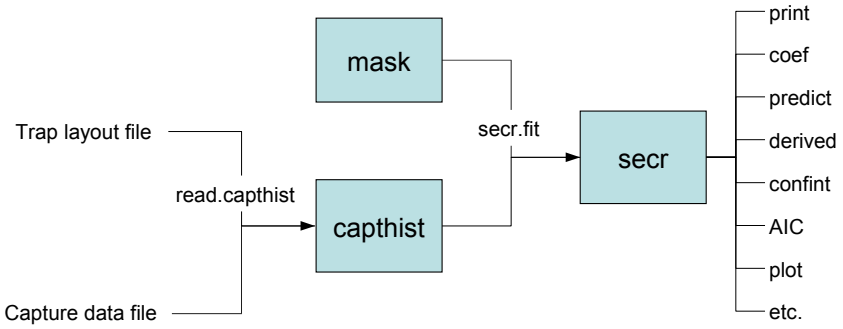


Figure 3: Essentials of the **secr** package. Each object class (shaded box) comes with methods to display and manipulate the data it contains (e.g. `print`, `summary`, `plot`, `rbind`, `subset`). The function `read.capthist` forms a `traps` object from the trap layout data and saves it as an attribute along with the capture data in a `capthist` object. If a habitat mask is not provided, one will be generated automatically by `secr.fit`. Any of the objects input to `secr.fit` may include a dataframe of covariates whose names may be used in a model formula. Fitted secr models may be further manipulated with the methods shown on the right. Additional functions (not shown) construct a regular detector array (e.g. `make.grid`, `make.circle`), form a mask from a `traps` object (`make.mask`), or simulate detection of a known population (`sim.capthist`).

## Output

The output from the function `secr.fit` is an object of class `secr`. This is an R list with many components. Assigning the output to a named object (such as secr0 or secrb in the example of Appendix 1) saves both the fit and the data for further manipulation. Typing the name at the R prompt invokes `print.secr` which formats the key results. These include the dataframe of estimates from the `predict` method for `secr` objects. Functions are provided for further computations on `secr` objects (e.g., profile-likelihood confidence intervals, AIC model selection, model averaging, likelihood-ratio and score tests). Many of these are listed in Appendix 3.

One system of units is used throughout `secr`. Distances are in metres and areas are in hectares (ha). The unit of density is animals per hectare. 1 ha = 10000 $m^2$ = 0.01 $km^2$. To convert density to animals / $km^2$, multiply by 100.

### Documentation

The primary documentation for **secr** is in the help pages that accompany the package. Help for a function is obtained in the usual way by typing a question mark at the R prompt, followed by the function name. Note the 'Index' link at the bottom of each help page – you will probably need to scroll down to find it.

The consolidated help pages are also distributed as the file `secr-manual.pdf`. Searching this text is a powerful way to locate a function for a particular task. It may be accessed from within R using

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

Other documentation in the form of pdf files based on Sweave vignettes will be added from time to time. The 'directory' link in the package help index lists available files. Each pdf file may be accessed by clicking on the link or with `RShowDoc()` as above. These vignettes are included in **secr** 2.5[3]:

| | |
|---|---|
| `secr-overview.pdf` | this document |
| `secr-datainput.pdf` | data formats and input functions |
| `secr-densitysurfaces.pdf` | modelling density surfaces |
| `secr-finitemixtures.pdf` | mixture models for individual heterogeneity |
| `secr-sound.pdf` | analysing data from microphone arrays |
| `secr-polygondetectors.pdf` | using polygon and transect detector types |
| `secr-varyingeffort.pdf` | variable effort in SECR models |

The web page http://www.otago.ac.nz/density/ should be checked for news of bug fixes and new releases. New versions will be posted on CRAN http://cran.r-project.org/, but there may be a delay of a few days. Help may be sought at http://www.phidot.org/forum; see also the FAQ there for DENSITY and **secr**. For information on changes in each version type

```
> news(package = "secr")
```

## Models in secr

Here 'models' relates to variation in the core SECR parameters that may be explained by known factors and covariates. Read Appendix 4 to make sense of this statement. If you just want to know how to use models, read on.

Models are defined symbolically in **secr** using R formula notation. A separate linear predictor is used for each core parameter. Core parameters are 'real'

[3]The Sweave Rnw files are available on request

parameters in the terminology of MARK, and **secr** uses that term because it will be familiar to biologists. Four real parameters are commonly modelled in **secr** 2.5; these are denoted D (for density), g0, sigma and z. Only the last three real parameters, which jointly define the model for detection probability as a function of location, can be estimated directly when the model is fitted by maximizing the conditional likelihood (`CL = TRUE` in `secr.fit`). D is then a derived parameter that is computed from an **secr** object with the function `derived` or one of its siblings (`derived.cluster etc.`). 'z' is a shape parameter that is used only when the detection function has three parameters (annular halfnormal, cumulative gamma, hazard-rate etc. – see `?detectfn`).

Detection parameters and density parameters are modelled separately, as we now describe.

## Specifying effects on detection parameters

Effects on parameters of detection probability are specified with R formulae. The variable names used in formulae are either names for standard effects (Table 4) or the names of user-supplied covariates. Effects 'b', 'B', 'bk', and 'Bk' refer to individuals whereas 'k' and 'K' refer only to sites. Groups ('g') are used only in models fitted by maximizing the full likelihood; for conditional likelihood models use a factor covariate to achieve the same effect.

Table 4: Automatically generated predictor variables used in detection models

| Variable | Description | Notes |
|---|---|---|
| g | group | interaction of the capthist individual covariates listed in argument `groups` of `secr.fit` |
| t | time factor | one level for each occasion |
| T | time trend | linear trend over occasions on link scale |
| b | learned response | step change after first detection |
| B | transient response | depends on detection at preceding occasion (Markovian response) |
| bk | animal x site response | site-specific step change |
| Bk | animal x site response | site-specific transient response |
| k | site learned response | site effectiveness changes once any animal caught |
| K | site transient response | site effectiveness depends on preceding occasion |
| session | session factor | one level for each session |
| Session | session trend | linear trend on link scale |
| h2 | 2-class mixture | finite mixture model with 2 latent classes |

Table 5: User-provided covariates used in detection models. The names of columns in the respective dataframes and names of components in the timevaryingcov attribute may be used in model formulae

| Covariate type | Data source | Notes |
|---|---|---|
| Individual | `covariates(capthist)` | conditional likelihood |
| Time | `timecov` argument | |
| Detector | `covariates(traps(capthist))` | |
| Detector x Time | `covariates(traps(capthist))` | see `timevaryingcov` |
| Session | `sessioncov` argument | |

Table 6: Some examples of the `model` argument in `secr.fit`

| Model | Description |
|---|---|
| `g0 ~ 1` | g0 is constant across animals, occasions and detectors |
| `g0 ~ b` | learned response affects g0 |
| `list(g0~b, sigma~b)` | learned response affects both g0 and sigma |
| `g0 ~ h2` | 2-class finite mixture for heterogeneity in g0 |
| `g0 ~ b + T` | learned response in g0 combined with trend over occasions |
| `sigma ~ g` | detection scale sigma differs between groups |
| `sigma ~ g*T` | group-specific trend in sigma |
| `D ~ cover` | density varies with 'cover' given in `covariates(mask)` |
| `list(D~g, g0~g)` | both density and g0 differ between groups |
| `D ~ session` | session-specific density |

Any name in a formula that is not a variable in Table 4 is assumed to refer to a user-supplied covariate. `secr.fit` looks for user-supplied covariates in data frames embedded in the `capthist` argument, or supplied in the `timecov` and `sessioncov` arguments, or named with the `timevaryingcov` attribute of a traps object, using the first match (Table 5).

The formula for any detection parameter (g0, sigma, z) may be constant (~1, the default) or some combination of terms in standard R formula notation (see `?formula`). For example, g0 ~ b + T specifies a model with a learned response and a linear time trend in g0; the effects are additive on the link scale. See Table 6 for other examples.

For other effects, the design matrix for detection parameters may also be provided manually in the argument `dframe` of `secr.fit`. This feature is untested.

## Density submodels

The SECR log likelihood is evaluated by summing values at points on a 'habitat mask' (the `mask` argument of `secr.fit`). Each point in a habitat mask represents a grid cell of potentially occupied habitat (their combined area may be almost any shape). The full design matrix for density (D) has one row for each point in the mask. As for the detection submodels, the design matrix has one column for the intercept (constant) term and one for each predictor.

Predictors may be based on Cartesian coordinates (e.g. 'x' for an east-west trend), a continuous habitat variable (e.g. vegetation cover) or a categorical (factor) habitat variable. Predictors must be known for all points in the mask (non-habitat excluded). The variables 'x' and 'y' are the coordinates of the habitat mask and are automatic, as are 'x2', 'y2', and 'xy'. Other spatial covariates should be named columns in the `covariates` attribute of the habitat mask.

See the vignette `secr-densitysurfaces.pdf` for more on fitting and displaying density surfaces.

## Model fitting and estimation

Models are fitted in `secr.fit` by numerically maximizing the likelihood. The likelihood involves integration over the unknown locations of the animals' range centres. This is achieved in practice by summation over points in the habitat mask, which has some implications for the user. Computation may be slow, especially if there are many points in the mask, and estimates may be sensitive to the particular choice of mask (either explicitly in `make.mask` or implicitly via the `buffer` argument).

The default maximization algorithm is Newton-Raphson in the function `stats::nlm`. By default, all reported variances, covariances, standard errors and confidence limits are asymptotic and based on a numerical estimate of the information matrix. The Newton-Raphson algorithm is fast, but it sometimes fails to compute the information matrix correctly, causing some standard errors to be set to 'NA'; see the 'method' argument of `secr.fit` for alternatives. Use `confint.secr` for profile likelihood intervals and `simulate.secr` for parametric bootstrap intervals (both are slow).

## Habitat masks

We have already introduced the idea of a habitat mask. The SECR likelihood is evaluated by summing values at points on a mask[4]; each point represents a grid cell of potentially occupied habitat. Masks may be constructed by placing a buffer of arbitrary width around the detectors, possibly excluding known non-habitat. How wide should the buffer be? The general answer is 'Wide enough not to cause bias in estimated densities'. This depends on the scale of movement of your animal, and on the chosen detection function. For specifics, see the help for `mask` and the various mask-related functions (`make.mask`, `mask.check`, `suggest.buffer`, and `esa.plot`). Heavy-tailed detection functions such as the hazard-rate and lognormal can be problematic because they require an unreasonably large buffer for stable density estimates.

## Varying effort

The probability of observing an individual at a particular detector may depend directly on a known quantity such as how long the detector was exposed on a particular occasion. In the extreme, a detector may not have been operated. The terms 'effort' and 'usage' are used here interchangeably for variation in the duration of exposure and similar known effects. Usage is an attribute of the detectors in a traps object (a traps x occasions matrix); it may be entered with the detector coordinates in a trap layout file or added later. Models fitted to data including a usage attribute will adjust automatically for varying usage across detectors and occasions. From secr 2.5.0 usage may take any non-negative value (previously binary). This simplifies the modelling of data aggregated over varying numbers of occasions or nearby sites.

See the separate document 'secr-varyingeffort.pdf' for more.

## Detector clusters

For surveying large areas it is efficient to use groups of detectors: within a group the detectors are close enough that animals may be re-detected at multiple points, while groups of detectors may be distributed across a region according to a probability design. From version 2.1 **secr** allows for detector groups with the 'cluster' data structure. This is an attribute of a traps object that records which detectors belong to which cluster[5].

Functions are provided to generate detector arrays with a clustered structure

[4] A 'mask' in **secr** is equivalent to a 'mesh' in DENSITY

[5] At present, clusters are assumed to share the same geometry (number of detectors, within-cluster spacing etc.)

(`trap.builder`, `make.systematic`), to extract or replace the cluster attribute (`clusterID`), to compute the geometric centres and numbers of detections per cluster (`cluster.centres`, `cluster.counts`), etc.

Data from a large, clustered design may often be analysed more quickly if the `capthist` object is first collapsed into one using the geometry of a single cluster (the object retains a memory of the number of individuals from each original cluster in the attribute `n.mash`). Use the function `mash` for this. Functions `derived`, `derived.mash` and the method `predict.secr` use `n.mash` to adjust their output densitiy, SE, and confidence limits.

### Parallel processing

From version 2.4.0 on it is possible to use multiple cores for certain computations. The greatest benefit is seen with simulations (`sim.secr`, `ip.secr`). See ?Parallel.

## References


Borchers, D. L., Buckland, S. T. and Zucchini, W. (2002) *Estimating animal abundance: closed populations*. Springer, London.

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

Cooch, E. and White, G. (eds) (2008) *Program MARK: A Gentle Introduction*. 6th edition. Available online at http://www.phidot.org/software/mark/docs/book/.

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

Efford, M. G. (2011) Estimation of population density by spatially explicit capture–recapture analysis of data from area searches. *Ecology* **92**, 2202–2207.

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

Efford, M. G. and Fewster, R. M. (2012) Estimating population size by spatially explicit capture–recapture. *Oikos* http://dx.doi.org/10.1111/j.1600-0706.2012.20440.x.

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, M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer. Pp 255–269.

Efford, M. G., Dawson, D. K. and Borchers, D. L. (2009) Population density estimated from locations of individuals on a passive detector array. *Ecology* **90**, 2676–2682.

Huggins, R. M. (1989) On the statistical analysis of capture experiments. *Biometrika* **76**, 133–140.

Laake, J. and Rexstad E. (2008) Appendix C. RMark - an alternative approach to building linear models in MARK. In: Cooch, E. and White, G. (eds) *Program MARK: A Gentle Introduction*. 6th edition. Available online at http://www.phidot.org/software/mark/docs/book/.

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* **62**.

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

Royle, J. A., Nichols, J. D., Karanth, K. U. and Gopalaswamy, A. M. (2009). A hierarchical model for estimating density in camera-trap studies. *Journal of Applied Ecology* **46**, 118–127.

Singh, P., Gopalaswamy, A. M., Royle, A. J., Kumar, N. S. and Karanth, K. U. (2010) *SPACECAP: A program to estimate animal abundance and density using Bayesian spatially explicit capture-recapture models. Version 1.0.* Wildlife Conservation Society - India Program, Centre for Wildlife Studies, Bangalure, India.

Stanley, T. R. and Burnham, K. P. (1999) A closure test for time-specific capture–recapture data. *Environmental and Ecological Statistics* **6**, 197–209.

# Appendix 1. A simple secr analysis

A simple analysis might look like this. We start by loading the package, setting the working folder, and constructing an object `myCH` that contains both the captures and the trap locations.

```
> library(secr)
> olddir <- setwd(system.file("extdata", package = "secr"))
> myCH <- read.capthist("capt.txt", "trap.txt", fmt = "XY")

No errors found :-)

> setwd(olddir)
```

Next we fit two simple models and compare them with AIC. We set `trace = FALSE` to reduce the volume of output, but the default `trace = TRUE` is usually better.

```
> secr0 <- secr.fit(myCH, model = g0 ~ 1, trace = FALSE)
> secrb <- secr.fit(myCH, model = g0 ~ b, trace = FALSE)
> AIC(secr0, secrb)

                model   detectfn npar    logLik      AIC     AICc
secr0 D~1 g0~1 sigma~1 halfnormal    3 -759.0198 1524.040 1524.373
secrb D~1 g0~b sigma~1 halfnormal    4 -759.0106 1526.021 1526.584
      dAICc  AICwt
secr0 0.000 0.7513
secrb 2.211 0.2487
```

A model with learned trap response (g0∼b) showed no improvement in fit over a null model (g0∼1). In this instance the estimates of density from the two models were also very close (not shown) and we rely on the null model for estimation. Before displaying the estimates we check that the likelihood is stable as we vary the mask buffer width (rows) and spacing (columns)

```
> mask.check(secr0)

Computing log likelihoods...
      spacing
buffer   7.34375 5.5078125  3.671875
   100 -759.0258 -759.0248 -759.0256
   150 -759.0165 -759.0165 -759.0165
   200 -759.0165 -759.0165 -759.0165
```

It seems we would have been better to use a buffer slightly wider than the default (100 m), so we repeat the fit and display the results:

```
> secr.fit(myCH, model = g0 ~ 1, buffer = 150, trace = FALSE)

secr.fit( capthist = myCH, model = g0 ~ 1, buffer = 150, trace =
  FALSE )
secr 2.5.0, 17:33:32 21 Jan 2013

Detector type     multi
Detector number   100
Average spacing   30 m
x-range           365 635 m
y-range           365 635 m
N animals       :  76
N detections    :  235
N occasions     :  5
Mask area       :  32.49 ha

Model           :  D~1 g0~1 sigma~1
Fixed (real)    :  none
Detection fn    :  halfnormal
Distribution    :  poisson
N parameters    :  3
Log likelihood  :  -759.0165
AIC             :  1524.033
AICc            :  1524.366

Beta parameters (coefficients)
            beta    SE.beta       lcl        ucl
D      1.7001486 0.11771578  1.469430  1.9308672
g0    -0.9785242 0.13620908 -1.245489 -0.7115593
sigma  3.3800087 0.04445186  3.292885  3.4671327

Variance-covariance matrix of beta parameters
                 D           g0        sigma
D      0.013857005  0.000184279 -0.001013493
g0     0.000184279  0.018552914 -0.003342478
sigma -0.001013493 -0.003342478  0.001975968

Fitted (real) parameters evaluated at base levels of covariates
       link   estimate SE.estimate       lcl        ucl
```

```
D       log  5.4747606  0.64670477  4.346756  6.8954877
g0    logit  0.2731847  0.02704497  0.223482  0.3292544
sigma   log 29.3710255  1.30624188 26.920407 32.0447288
```

The density estimate is 5.475 ha$^{-1}$ (95% confidence interval 4.35–6.90 ha$^{-1}$). We can compare these estimates to those from the initial fit with a narrower buffer; estimated density differs only in the third decimal place:

```
> predict(secr0)

       link   estimate SE.estimate        lcl        ucl
D       log  5.4788228  0.64671178  4.3507097  6.8994490
g0    logit  0.2731605  0.02705172  0.2234467  0.3292454
sigma   log 29.3695687  1.30602218 26.9193496 32.0428087
```

# Appendix 2. Software feature comparisons

• full implementation; ◦ incomplete or inferior implementation.

| Feature | DENSITY 5.0 | **secr** 2.0 | **secr** 2.5 |
|---|---|---|---|
| *General* | | | |
| Graphical interface | • | ◦ | ◦ |
| Inverse prediction (IP SECR) | • | • | • |
| Maximum likelihood estimation (ML SECR) | • | • | • |
| Non-spatial closed-population estimators | | • | • |
| Simulation of spatial sampling | • | ◦ | ◦ |
| Build detector arrays | • | ◦ | • |
| Control of random number generator | ◦ | • | • |
| Closure tests | ◦ | • | • |
| Import or export DENSITY text files | • | • | • |
| Import or export SPACECAP text files | | • | • |
| Convert BUGS data | | ◦ | ◦ |
| GIS polygons as habitat mask | • | • | • |
| Clustered detector layouts | | | • |
| Mash data from clustered layouts | | | • |
| Upload coordinates to GPS (uses GPSBabel) | | | • |
| Multi-core processing | | | ◦ |
| *ML SECR* | | | |
| Profile likelihood confidence intervals | • | • | • |
| Varying effort (detector usage) | ◦ | ◦ | • |
| Fixed parameters | ◦ | • | • |
| Parametric bootstrap | ◦ | • | • |
| Between-session models | • | • | • |
| Mixture models for individual heterogeneity | • | • | • |
| Confidence ellipses | • | • | • |
| Formula-based model notation | | • | • |
| Density models (inhomogeneous 2-D Poisson) | | • | • |
| User-defined density models | | | • |
| Plot density models | | | • |
| Groups (e.g. males & females) | | • | • |
| Score tests for model selection | | • | • |
| Model averaging | | • | • |
| Structural relationships between parameters | | • | • |

| | | | |
|---|---|---|---|
| Plot likelihood surface | | • | • |
| Empirical variance from replicate units | | • | • |
| Mask diagnostics | ○ | • | • |
| Suggested buffer width | | • | • |
| Contours of detection probability | • | • | • |
| Compute pdf for individual's range centre | • | • | • |
| Regional population size (region.N) | | | • |
| Time-varying detector covariates | | | • |
| Combined telemetry-detection models | | | ○ |
| *Detector types* | | | |
| Single-catch trap[1] | ○ | ○ | ○ |
| Multi-catch trap | • | • | • |
| Proximity | • | • | • |
| Signal strength (acoustic) | | • | • |
| Count | | • | • |
| Polygon | | • | • |
| Transect | | • | • |
| Polygon (exclusive) | | • | • |
| Transect (exclusive) | | • | • |
| Telemetry | | | • |
| Unmarked | | | ○ |
| Presence/absence | | | ○ |
| *Detection functions* | | | |
| Halfnormal | • | • | • |
| Hazard rate[2] | • | • | • |
| Exponential | • | • | • |
| Compound halfnormal | | • | • |
| Uniform[1] | ○ | ○ | ○ |
| w-exponential | | • | • |
| Annular halfnormal | | • | • |
| Binary signal strength | | • | • |
| Signal strength | | • | • |
| Signal strength spherical | | • | • |
| Cumulative lognormal[2] | | • | • |
| Cumulative gamma | | • | • |

[1]Not fitted by ML SECR
[2]Not recommended because of heavy tail

# Appendix 3. Functions in secr arranged according to use

This list groups the main functions of **secr** 2.5. Many functions for data manipulation and plotting are omitted. S3 methods are marked with an asterisk *

*Manipulate core objects*

| | |
|---|---|
| `addCovariates` | add spatial covariates to `traps` or `mask` |
| `head`* | first rows of `capthist`, `traps` or `mask` |
| `join` | combine sessions of multi-session `capthist` object |
| `make.grid` | construct detector array |
| `make.capthist` | form `capthist` from `traps` and detection data |
| `make.mask` | construct habitat mask (mesh) |
| `make.systematic` | construct random systematic design |
| `MS.capthist` | combine `capthist` objects into one multisession `capthist` |
| `plot`* | plot `capthist`, `traps` or `mask` |
| `rbind.capthist` | append `capthist` objects |
| `read.capthist` | input captures and trap layout from Density format, one call |
| `read.traps` | input detector locations from text file |
| `reduce`* | aggregate nearby detectors or occasions |
| `sim.capthist` | simulate capture histories |
| `subset`* | filter `capthist`, `traps` or `mask` |
| `snip` | split transect(s) into equal sections |
| `summary`* | summarise `capthist`, `traps` or `mask` |
| `tail`* | last rows of `capthist`, `traps` or `mask` |
| `trap.builder` | construct various complex designs |
| `verify`* | check `capthist`, `traps` or `mask` for internal consistency |
| `randomHabitat` | generates habitat mask with random landscape |

*Extract or replace attributes of* `traps` *object*

| | |
|---|---|
| `clusterID` | cluster identifier |
| `clustertrap` | detector number within cluster |
| `covariates`* | detector-level covariates |
| `detector`* | detector type ('multi', 'proximity' etc.) |
| `polyID`* | polygon or transect identifier |
| `timevaryingcov` | name time-varying covariate(s) |
| `usage`* | occasion- and detector-specific effort |

*Extract or replace attributes of* `capthist` *object*

| | |
|---|---|
| `addTelemetry` | add telemetry data to a 'proximity' or 'count' object |

| | |
|---|---|
| `covariates`* | individual-level covariates, including grouping factors |
| `session`* | session identifier(s) |
| `session`* | session identifier(s) |
| `signalmatrix` | sound x microphone table |
| `traps`* | embedded `traps` object(s) |

*Extract detection-specific data from* `capthist` *object*

| | |
|---|---|
| `alive` | TRUE/FALSE |
| `animalID` | individual ID |
| `clusterID` | cluster identifier |
| `clustertrap` | detector number within cluster |
| `noise` | noise (signal detectors) |
| `occasion` | occasion |
| `signal` | signal strength (signal detectors) |
| `signalframe` | whole signal & noise dataframe (rows = detections) |
| `trap` | detector |
| `xy` | detection coordinates (polygon and transect detectors) |

*Fit SECR model*

| | |
|---|---|
| `ip.secr` | fit simple SECR model by simulation & inverse prediction |
| `secr.fit` | maximum likelihood fit; result is a fitted `secr` object |

*Operate on fitted* `secr` *object(s)*

| | |
|---|---|
| `AIC`* | model selection, model weights |
| `coef`* | 'beta' parameters |
| `collate` | tabulate estimates from several models |
| `confint`* | profile likelihood confidence intervals |
| `derived` | density from conditional likelihood models |
| `deviance`* | model deviance |
| `df.residual`* | degrees of freedom for deviance |
| `derived.nj` | variance from replicated sampling units |
| `derived.cluster` | variance from replicated sampling units |
| `derived.external` | variance from replicated sampling units |
| `fxi.secr` | probability density of home-range centre |
| `logLik`* | log-likelihood of fitted model |
| `LR.test` | likelihood-ratio test of two models |
| `model.average` | combine estimates using AICc weights |
| `plot`* | plot detection functions with confidence bands |
| `predict`* | 'real' parameters for arbitrary levels of predictor variables |
| `predictDsurface`* | evaluate density surface at each point of a mask |
| `score.test` | model selection with score statistic using observed information |

| | |
|---|---|
| `simulate`* | generate realisations of fitted model |
| `sim.secr` | parametric bootstrap |
| `vcov`* | variance-covariance matrix of 'beta' or 'real' parameters |

*Mask diagnostics*

| | |
|---|---|
| `suggest.buffer` | find buffer width to keep bias within bounds |
| `esa.plot` | cumulative plot esa vs buffer width |
| `mask.check` | likelihood or estimates vs. buffer width and spacing |

*Specialised graphics*

| | |
|---|---|
| `fxi.contour` | contour plot of home-range centre pdf(s) |
| `pdot.contour` | contour plot of detection probability |
| `buffer.contour` | concave and convex boundary strips |

*Convert or export data*

| | |
|---|---|
| `RMarkInput` | convert `capthist` to dataframe for RMark |
| `write.capthist` | export `capthist` as text files for DENSITY |
| `write.DA` | convert `capthist` for analysis in WinBUGS |
| `write.SPACECAP` | export `capthist` as text files for SPACECAP |
| `writeGPS` | upload coordinates to GPS using GPSBabel |

*Miscellaneous*

| | |
|---|---|
| `ARL` | asymptotic range length |
| `autoini` | generate starting values of D, g0 and sigma for `secr.fit` |
| `closure.test` | closure tests of Otis et al. (1978) and Stanley & Burnham (1999) |
| `closedN` | closed population size by various conventional estimators |
| `counts` | summary data from `capthist` object |
| `dbar` | mean distance between capture locations |
| `distancetotrap` | from an arbitrary set of points |
| `MMDM` | mean maximum distance moved |
| `moves` | distances between capture locations |
| `nearesttrap` | from an arbitrary set of points |
| `pdot` | location-specific net probability of detection |
| `RPSV` | 'root pooled spatial variance', a simple measure of home-range size |

*Datasets – see `?datasetname` for details*

| | |
|---|---|
| `deermouse` | *Peromyscus maniculatus* live-trapping data of V. H. Reid published as a CAPTURE example by Otis et al. (1978) *Wildlife Monographs* **62** |

| | |
|---|---|
| `hornedlizard` | repeated searches of a quadrat in Arizona for flat-tailed horned lizards *Phrynosoma mcallii* (Royle & Young *Ecology* **89**, 2281–2289) |
| `housemouse` | *Mus musculus* live-trapping data of H. N. Coulombe published as a CAPTURE example by Otis et al. (1978) *Wildlife Monographs* **62** |
| `ovenbird` | multi-year mist-netting study of ovenbirds *Seiurus aurocapilla* at a site in Maryland, USA. |
| `ovensong` | acoustic detections of ovenbirds (Dawson & Efford *Journal of Applied Ecology* **46**, 1201–1209) |
| `possum` | brushtail possum *Trichosurus vulpecula* live trapping at Waitarere, North Island, New Zealand April 2002 (Efford et al. 2005 *Wildlife Society Bulletin* **33**, 731–738) |
| `secrdemo` | simulated data `captdata` and some fitted models |
| `skink` | multi-session lizard (*Oligosoma infrapunctatum* and *O. lineoocellatum*) pitfall trapping data from Lake Station, Upper Buller Valley, South Island, New Zealand (M. G. Efford, B. W. Thomas and N. J. Spencer unpublished). |
| `stoatDNA` | stoat *Mustela erminea* hair tube DNA data from Matakitaki Valley, South Island, New Zealand (Efford, Borchers and Byrom 2009). |

# Appendix 4. Models in secr

A family of capture–recapture models, such as the Cormack-Jolly-Seber models for survival, may include submodels[6] that allow for variation in core ('real') parameters, including the effects of covariates. Annual survival, for example, may vary with the severity of winter weather, so it often makes sense to include a measure of winter severity as a covariate. Gary White's MARK software has been particularly successful in packaging open-population models for biologists, and **secr** aims for similar flexibility.

The language of generalised linear models is convenient for describing submodels (e.g. Huggins 1989, Lebreton et al. 1992). Each parameter is treated as a linear combination of predictor variables on its transformed ('link') scale. This is useful for combining effects because, given a suitable link function, any combination maps to a feasible value of the parameter. The logit scale has this property for probabilities in (0, 1), and the natural log scale works for positive parameters i.e. $(0, +\infty)$. These are the link functions used most often in **secr**, but there are others, including the identity (null) link.[7]

Submodels are defined symbolically in **secr** using R formula notation. A separate linear predictor is used for each core parameter. Core parameters are 'real' parameters in the terminology of MARK, and **secr** uses that term because it will be familiar to biologists. Four real parameters are commonly modelled in **secr** 2.5; these are denoted D (for density), g0, sigma and z. Only the last three real parameters, which jointly define the model for detection probability as a function of location, can be estimated directly when the model is fitted by maximizing the conditional likelihood (`CL = TRUE` in `secr.fit`). D is then a derived parameter that is computed from an `secr` object with the function `derived` or one of its siblings (`derived.cluster etc.`). 'z' is a shape parameter that is used only when the detection function has three parameters (annular halfnormal, cumulative gamma, hazard-rate etc. – see `?detectfn`).

For each real parameter there is a linear predictor of the form $\mathbf{y} = \mathbf{X}\beta$, where $\mathbf{y}$ is a vector of parameter values on the link scale, $\mathbf{X}$ is a design matrix of predictor values, and $\beta$ is a vector of coefficients. Each element of $\mathbf{y}$ and corresponding row of $\mathbf{X}$ relates to the value of the real parameter in a particular circumstance (e.g. density at a particular point in space, or detection probability of an animal on a particular occasion). The elements of $\beta$ are coefficients estimated when we fit the model. In MARK these are called 'beta parameters' to distinguish them from the transformed 'real' parameter values in $\mathbf{y}$. **secr** acknowledges this usage, but also refers to beta parameters as 'coefficients' and real parameters as 'fitted values', a usage more in line with other statistical

[6] This use of 'submodel' is non-standard – maybe we'll find a better term

[7] Set link functions with the `link` argument of `secr.fit`.

modelling in R. $\mathbf{X}$ has one column for each element of $\beta$. Design matrices are described in more detail in the next section.

## Design matrices

A design matrix is specific to a 'real' parameter. Each design matrix $\mathbf{X}$ contains a column of '1's (for the constant or intercept term) and additional columns as needed to describe the effects in the submodel for the parameter. Depending on the model, these may be continuous predictors (e.g. air temperature to predict occasion-to-occasion variation in g0), indicator variables (e.g. 1 if animal $i$ was caught before occasion $s$, 0 otherwise), or coded factor levels. Within `secr.fit`, each design matrix is constructed automatically from the input data and the model formula in a 2-stage process.

First, a data frame is built containing 'design data' with one column for each variable in the formula. Second, the R function `model.matrix` is used to construct the design matrix. This process is hidden from the user. The design matrix will have at least one more column than the design data; there may be more if the formula includes interactions or factors with more than two levels. For a good description of this general approach see the documentation for RMark (Laake and Rexstad 2008). The necessary design data are either extracted from the inputs or generated automatically, as explained in later sections. 'Real' parameters fall into two groups: density (D) and detection (g0, sigma and z). Density and detection parameters are subject to different effects, so they use different design matrices as described in the next three sections.

## Detection submodels

For SECR, we want to model the detection of each individual $i$ on occasion $s$ at detector $k$. Given $n$ observed individuals on $S$ occasions at $K$ detectors, there are therefore $nSK$ detection probabilities of interest. We treat these as elements in a 3-dimensional array. Strictly, we are also interested in the detection probabilities of unobserved individuals, but these are estimated only by extrapolation from those observed so we do not include them in the array.

In a null model, all $nSK$ detection probabilities are assumed to be the same. The conventional sources of variation in capture probability (Otis et al. 1978) appear as variation either in the $n$ dimension ('individual heterogeneity' h), or in the $S$ dimension ('time variation' t), or as a particular interaction in these two dimensions ('behavioural response to capture' b). Combined effects are possible. SECR introduces additional complexity.

Detection probability in SECR is no longer a scalar (even for a particular

animal-occasion-detector combination); it is described by a 'detection function'. The detection function may have two parameters (e.g. g0, sigma for a half-normal function), or three parameters (e.g. g0, sigma, z). Any of the parameters of the detection function may vary with respect to individual (subscript $i$), occasion (subscript $s$) or detector (subscript $k$).

The full design matrix for each detection submodel has one row for each combination of $i$, $s$ and $k$. Allowing a distinct probability for each animal (the $n$ dimension) may seem excessive, and truly individual-specific covariates are feasible only when a model is fitted by maximizing the conditional likelihood (cf Huggins 1989). However, the full $nSK$ array is convenient for coding both group membership (Lebreton et al. 1992, Cooch and White 2008) and experience of capture, even when individual-specific covariates cannot be modelled.

The programming gets even more complex. Analyses may combine data from several independent samples, dubbed 'sessions'. This adds a fourth dimension of length equal to the number of sessions. When finite mixture models are used for detection parameters there is even a fifth dimension, with the preceding structure being replicated for each mixture class. Fortunately, **secr** handles all this out of view: as a user you only need to know how to specify the detection model.