secr 2.9 - spatially explicit capture–recapture in R

Murray Efford

2015-06-14

Contents

This document provides an overview of **secr** 2.9, 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 (this is a good place to start if you are new to **secr**). For details of how to use **secr** see the help pages and vignettes.

Two add-on packages extend the capability of **secr** and are documented separately. **secrlinear** enables the estimation of linear density (e.g., animals per km) for populations in linear habitats such as stream networks (secrlinear-vignette.pdf). **secrdesign** enables the assessment of alternative study designs by Monte Carlo simulation; scenarios may differ in detector (trap) layout, sampling intensity, and other characteristics (secrdesign-vignette.pdf).

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'. The methods are used primarily to estimate population density, but they also have advantages over non-spatial methods when the goal is to estimate population size (Efford and Fewster 2013). 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 area (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).

Table 1. Some 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.

	Occasion				
ID	1	2	3	4	5
1	A9	.	.	.	.
2	A12	A12	.	.	.
3	.	.	C6	B5	.
4	.	.	G3	.	F3
etc.					

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 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, Royle and Gardner 2011, Royle et al. 2014), but this approach is much slower than ML SECR; 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 (= intensity) is the sole parameter of a homogeneous Poisson process. An inhomogeneous Poisson distribution may also be fitted; this provides a means to evaluate the effects of habitat variables on density.

```
library(secr)
```

```
par(mfrow=c(1,1), pty='s', mar=c(2,2,2,2), cex=1.2)
plot(sim.popn(D=5, core=make.grid(), buffer=150), cex=1)
plot(make.grid(),add=T, detpar=list(col = "red", pch = 3, cex = 1.2))
```

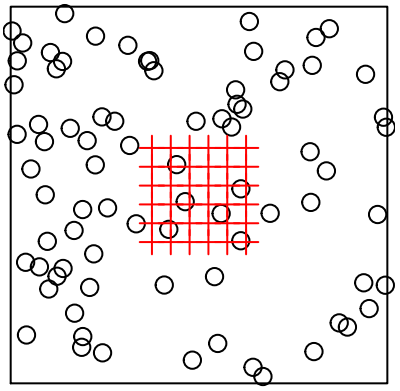

Fig. 1. Hypothetical Poisson distribution of range centres near an array of detectors. Each dot represents one individual. SECR estimates the density of this distribution.

Detection functions

A detection model describes the decline in detection probability with distance (d) from the home-range centre (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).

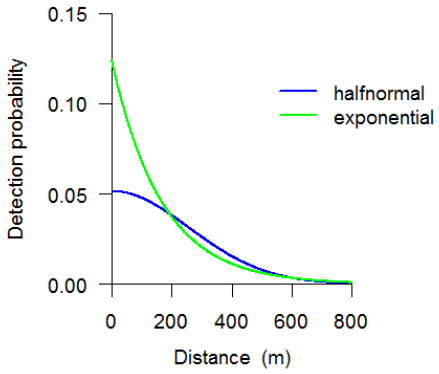

Fig. 2. Alternative detection functions. The halfnormal is defined by $g(d) = g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right)$ and the exponential by $g(d) = g_0 \exp\left(-\frac{d}{\sigma}\right)$. See ?detectfn for more (also the list in Appendix 2).

Detector types

The properties of detectors are an important part of the SECR observation model (Table 2). Inside **secr**, data are tagged with a detector type to ensure they are printed, plotted and analysed appropriately.

Some common detectors (camera 'traps' and hair snags for DNA) do not capture animals, but merely record that an animal has visited a site. 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 (for such binary proximity detectors each observation has a Bernoulli distribution). 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 via the 'binomN' argument of `secr.fit` (binomN = 0 indicates Poisson; binomN > 1 indicates binomial with size = binomN; binomN = 1 indicates binomial with size given by the 'usage' attribute for the detector and occasion).

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 some support in **secr** 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 2. Detector types in **secr**

Detector	Description
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), Efford (2011), Efford and Fewster (2013), Efford et al. (2013) and Efford and Mowat (2014). **secr** uses external C code for computationally intensive operations. Appendix 2 compares the features of DENSITY and **secr**. The most 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 and models fitted to those data.

Table 3. Essential classes in **secr**.

Class	Data
traps	locations of detectors; detector type ('proximity', 'multi', etc.)
capthist	spatial detection histories, including a 'traps' object
mask	raster map of habitat near the detectors
secr	fitted SECR model

To perform an SECR analysis you explicitly or implicitly construct each of these objects in turn. Fig. 3 summarizes the relationships among the classes.

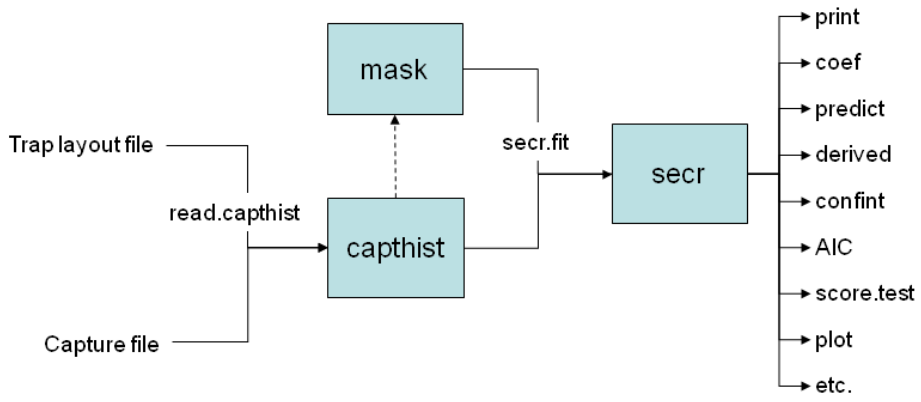

Fig. 3. Essentials of the **secr** package.

- Each object class (shaded box) comes with methods to display and manipulate the data it contains (e.g. `print`, `summary`, `plot`, `rbind`, `subset`)[2].
- The function `read.capthist` forms a 'traps' object from the detector layout data and saves it as an attribute, along with capture data read from another file, in a 'capthist' object.
- By default, a habitat mask is generated automatically by `secr.fit` using a specified buffer around the detectors (traps). The function `make.mask` gives greater control over this step.

[1] Technically, these are S3 classes. A 'class' in R 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 this font refers to R objects that are documented in online help for the **secr** package, or in base R.

- Any of the objects input to `secr.fit` (traps, capthist, mask) may include a dataframe of covariates saved as an attribute. Covariate names may be used in model formulae; the `covariates` method is used to extract or replace covariates.
- Fitted secr models may be further manipulated with the methods shown on the right and others listed in Appendix 4.

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. It is also possible 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 more general route may be needed for unusual datasets.

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., AIC model selection, model averaging, profile-likelihood confidence intervals, and likelihood-ratio tests). Many of these are listed in Appendix 4.

One system of units is used throughout **secr**. Distances are in metres and areas are in hectares (ha). The unit of density for 2-dimensional habitat is animals per hectare. 1 ha = 10000 m^2 = 0.01 km^2. To convert density to animals per km^2, multiply by 100. Density in linear habitats (see package **secrlinear**) is expressed in animals per km.

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 index may also be accessed with `help(package = secr)`.

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 vignettes built with **knitr**, will be added from time to time. The 'User guides...' link in the package help index lists available files. The vignettes in Table 4 are included in **secr** 2.9 or may be found on the Density website.

Table 4. Vignettes for **secr** 2.9.

Vignette	Topic
secr-overview.pdf	introduction (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-noneuclidean.pdf	non-Euclidean distance models
secr-parameterisations.pdf	alternative parameterisations of detection
secr-polygondetectors.pdf	using polygon and transect detector types
secr-sound.pdf	analysing data from microphone arrays
secr-varyingeffort.pdf	variable effort (usage) 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, but there may be a delay of a few days. Help may be sought at phidot; see also the FAQ there for DENSITY and **secr**. Another forum intended for both software issues and wider discussion is secrgroup. For information on changes in each version, type at the R prompt:

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

Defining models with the 'model' argument of secr.fit

By default, the parameters of SECR models are assumed to be constant. We specify more interesting, and often better-fitting, models with the 'model' argument of `secr.fit`. Here 'models' relates to variation in the parameters that may be explained by known factors and covariates. Read Appendix 6 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' parameters in the terminology of MARK, and **secr** uses that term because it will be familiar to biologists.

Three real parameters are commonly modelled in **secr** 2.9; these are denoted 'D' (for density), 'g0' (or 'lambda0') and 'sigma'. Only the last two 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.).

Here is a simple example of the model argument in use (see also Appendix 1)

```
secr.fit(captdata, model = g0~t)
```

The real parameter g0 is no longer constant, but takes a unique value on each sampling occasion (t).

Other 'real' parameters appear in particular contexts. '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`). Some detection functions primarily model 'exposure' or the cumulative hazard of detection, rather than the probability of detection; these use the real parameter 'lambda0' in place of 'g0' (see ?detectfn). 'lambda0' is also used with count detectors. A further 'real' parameter is the mixing proportion 'pmix', used in finite mixture models and hybrid mixture models (see ?hcov).

Sometimes it is illuminating and efficient to parameterise the detection function using a function of the primary 'real' parameters described above. This gives rise to the surrogate 'real' parameters a0 and sigmak; see the vignette secr-parameterisations.pdf for details and references.

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

Detection parameters

Effects on parameters of detection probability are specified via R formulae. The variable names used in formulae are either names for standard effects (Table 5) 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. See also the later section on modelling sex differences.

Table 5. Automatically generated predictor variables used in detection models

Variable	Description	Notes
g	group	individual covariates listed in `secr.fit` argument 'groups'
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

Any name in a formula that is not a variable in Table 5 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 6).

Table 6. Types of User-provided covariate for 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 only
Time	timecov argument	
Detector	covariates(traps(capthist))	
Detector x Time	covariates(traps(capthist))	see ?timevaryingcov
Session	sessioncov argument	

The formula for any detection parameter (e.g., g0, lambda0 or sigma) may be constant ($\sim$ 1, the default) or some combination of terms in standard R formula notation (see ?formula). For example, g0 $\sim$ 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 7 for other examples.

Table 7. Some examples of the 'model' argument in `secr.fit`

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

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.

Inhomogeneous density models

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.

Regression splines are particularly effective for modelling spatial trend. For these and general guidance on fitting and displaying density surfaces, see the vignette secr-densitysurfaces.pdf.

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, but note the ncores argument of `simulate.secr`).

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[3]; 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 the 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.

Miscellaneous topics

Modelling sex differences

There are many ways to model sex differences in **secr**. Here we sketch some possibilities, in order of usefulness (your mileage may vary).

1. Fit a hybrid mixture model as described in the online help (?hcov). This accommodates occasional missing values and estimates the sex ratio (pmix).
2. Use conditional likelihood (`CL = TRUE`) and include a categorical (factor) covariate in model formulae (e.g., g0 $\sim$ sex). To get sex-specific densities then specify `groups = "sex"` in `derived`.
3. Use full likelihood (`CL = FALSE`) and separate data for the two sexes as different sessions (most easily, by coding 'female' or 'male' in the first column of the capture file read with `read.capthist`). Then include a group term 'session' in relevant model formulae (e.g., g0 $\sim$ session).
4. Use full likelihood (`CL = FALSE`), define `groups = "sex"` or similar, and include a group term 'g' in relevant formulae (e.g., g0 $\sim$ g).

'CL' and 'groups' are arguments of `secr.fit`. Possibilities 1–4 should not be mixed for comparing AIC. Sex differences in home-range size (and hence sigma) may be mitigated by compensatory variation in g0 or lambda0 (Efford and Mowat 2014).

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 (see ?usage). Models fitted to data including a usage attribute will adjust automatically for varying usage across detectors and occasions. 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 and Efford et al. (2013) for more.

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

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 to sample possible spatial variation in density. **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[4].

Functions are provided to generate detector arrays with a clustered structure (`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 density, SE, and confidence limits.

Parallel processing

It is possible to use multiple cores to speed up certain computations. The greatest benefit is seen with simulations (`sim.secr`, `ip.secr`) (see ?Parallel). The functions `par.secr.fit`, `par.region.N` and `par.derived` allow a collection of models to be fitted or analysed simultaneously using multiple cores.

Regression splines

The standard models for 'real' parameters in **secr** are linear on the link scale, much like a generalised linear model. For more flexibility is possible to use semi-parametric 'regression spline' smooths. These are implemented in **secr** using a method suggested by Borchers and Kidney (in prep.): Simon Wood's R package **mgcv** is used to parse s() and te() terms in model formulae and construct basis functions that are used like linear covariates within **secr**. Any 'real' parameter may be modelled with regression splines (D, lambda0, sigma, noneuc etc.). For details see the help page (?smooths) and the documentation for **mgcv**.

Non-Euclidean distances

'Distance' in SECR models usually, and by default, means the Euclidean distance $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. The observation model can be customised by replacing the Euclidean distance with one that 'warps' space in some ecologically meaningful way. There are innumerable ways to do this. Royle et al. (2013) envisioned an 'ecological distance' that is a function of landscape covariates. Redefining distance is a way to model spatial variation in the size of home ranges, and hence the spatial scale of movement σ; Efford et al. (in review) use this to model inverse covariation between density and home range size. Distances measured along a linear habitat network such as a river system are also non-Euclidean (see package **secrlinear**).

secr provides general tools for specifying and modelling non-Euclidean distance, via the `secr.fit` details component 'userdist'. This may be a user-specified function or a pre-computed matrix. See secr-noneuclidean.pdf for a full explanation and examples.

References

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

[4]Clusters are assumed to share the same geometry (number of detectors, within-cluster spacing etc.).

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) (2014) *Program MARK: A Gentle Introduction*. 13th 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., 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., Borchers D. L. and Mowat, G. (2013) Varying effort in capture–recapture studies. *Methods in Ecology and Evolution* **4**, 629–636.

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.

Efford, M. G., Dawson, D. K., Jhala, Y. V. and Qureshi, Q. In review. Density-dependent home-range size revealed by spatially explicit capture-recapture.

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

Efford, M. G. and Mowat, G. (2014) Compensatory heterogeneity in spatially explicit capture–recapture data. *Ecology* **95**, 1341–1348.

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

Laake, J. and Rexstad E. (2014) Appendix C. RMark - an alternative approach to building linear models in MARK. In: Cooch, E. and White, G. (eds) *Program MARK: A Gentle Introduction*. 13th edition. 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., Chandler, R. B., Gazenski, K. D. and Graves, T. A. (2013) Spatial capture–recapture models for jointly estimating population density and landscape connectivity. *Ecology* **94** 287–294.

Royle, J. A., Chandler, R. B., Sollmann, R. and Gardner, B. (2014) *Spatial capture–recapture.* Academic Press.

Royle, J. A. and Gardner, B. (2011) Hierarchical spatial capture–recapture models for estimating density from trapping arrays. In: A.F. O'Connell, J.D. Nichols and K.U. Karanth (eds) *Camera Traps in Animal Ecology: Methods and Analyses*. Springer, Tokyo. Pp. 163–190.

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.

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

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, Bangalore, 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. The file capt.txt from DENSITY uses an old data format 'XY' in which each detection has x-y coordinates that must be matched to x-y coordinates in trap.txt. The more usual format is 'trapID', which matches the detector identifier.

```
library(secr)                                                  # load package
setwd(system.file('extdata', package='secr'))                  # set working folder
myCH <- read.capthist('capt.txt','trap.txt', fmt = 'XY')       # import data using XY format
```

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

Next we fit two simple models and compare them with AIC. We set `trace = FALSE` to reduce the volume of output. The warning reminds us to check the buffer width, which we do later.

```
secr0 <- secr.fit(myCH, model = g0~1, buffer = 100, trace = FALSE)  # null model
secrb <- secr.fit(myCH, model = g0~b, buffer = 100, trace = FALSE)  # trap response model
AIC (secr0, secrb)                                                  # compare
```

```
##                  model   detectfn npar   logLik     AIC    AICc dAICc AICcwt
## secr0 D~1 g0~1 sigma~1 halfnormal    3 -759.026 1524.05 1524.38 0.000 0.7513
## secrb D~1 g0~b sigma~1 halfnormal    4 -759.016 1526.03 1526.60 2.211 0.2487
```

A model with learned trap response (g0 $\sim$ b) showed no improvement in fit over a null model (g0 $\sim$ 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
##   98.488  -759.028  -759.028 -759.027
##   147.732 -759.017  -759.017 -759.017
##   196.976 -759.017  -759.017 -759.017
```

It seems we would have been better to use a slightly wider buffer, 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 2.9.5, 08:49:43 14 Jun 2015
##
## 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       :  30.5546 ha
##
## Model           :  D~1 g0~1 sigma~1
## Fixed (real)    :  none
## Detection fn    :  halfnormal
## Distribution    :  poisson
## N parameters    :  3
## Log likelihood  :  -759.017
## AIC             :  1524.03
## AICc            :  1524.37
##
## Beta parameters (coefficients)
##            beta   SE.beta      lcl       ucl
## D      1.700149 0.1177157  1.46943  1.930867
## g0    -0.978524 0.1362091 -1.24549 -0.711559
## sigma  3.380008 0.0444517  3.29288  3.467132
##
## Variance-covariance matrix of beta parameters
##                   D           g0       sigma
## D      0.013856996  0.000184266 -0.00101348
## g0     0.000184266  0.018552923 -0.00334247
## sigma -0.001013482 -0.003342466  0.00197596
##
## Fitted (real) parameters evaluated at base levels of covariates
##        link  estimate SE.estimate       lcl       ucl
## D       log  5.474762    0.646705  4.346758  6.895489
## g0    logit  0.273185    0.027045  0.223482  0.329254
## sigma   log 29.371017    1.306238 26.920406 32.044712
```

The density estimate is 5.475 / ha (95% confidence interval 4.35–6.90 / ha). 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  5.479804   0.6467408  4.351623  6.900473
## g0    logit  0.273191   0.0270513  0.223477  0.329274
## sigma   log 29.365832   1.3049380 26.917571 32.036771
```

Appendix 2. Software feature comparisons

- • full implementation
- ◦ incomplete or inferior implementation.

Feature	DENSITY 5.0	**secr** 2.9
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		
Density models (inhomogeneous 2-D Poisson)		•
Regional population size (region.N)		•
Varying effort (detector usage)	◦	•
Fixed parameters	◦	•
Parametric bootstrap	◦	•
Between-session models	•	•
Profile likelihood confidence intervals	•	•
Mixture models for individual heterogeneity	•	•
Confidence ellipses	•	•
Formula-based model notation		•
Plot density models		•
Groups (e.g. males & females)		•
Score tests for model selection		•
Model averaging		•
Plot likelihood surface		•

Feature	DENSITY 5.0	**secr** 2.9
Empirical variance from replicate units		•
Mask diagnostics	◦	•
Suggested buffer width		•
Contours of detection probability	•	•
Compute pdf for individual's range centre	•	•
Time-varying detector covariates		•
Hybrid mixture models (hcov)		•
Compensation (a0 parameterization)		•
Density-dependent sigma (sigmak parameterization)		•
Variance-only mode (method = 'none')		•
Combined telemetry-detection models		•
Regression splines		•
Non-Euclidean distance		•
Detector types		
Single-catch trap[a]	◦	◦
Multi-catch trap	•	•
Proximity	•	•
Signal strength (acoustic)		•
Count		•
Polygon		•
Transect		•
Polygon (exclusive)		•
Transect (exclusive)		•
Telemetry		•
Unmarked		◦
Presence/absence		◦
Detection functions		
Halfnormal	•	•
Hazard rate[b]	•	•
Exponential	•	•
Compound halfnormal		•
Uniform[a]	◦	◦
w-exponential		•
Annular halfnormal		•
Binary signal strength		•
Signal strength		•

Feature	DENSITY 5.0	**secr** 2.9
Signal strength spherical		•
Cumulative lognormal[b]		•
Cumulative gamma		•
Hazard halfnormal		•
Hazard hazard rate[b]		•
Hazard exponential		•
Hazard annular halfnormal		•
Hazard cumulative gamma		•

a. Not fitted by ML secr

b. Not recommended because of heavy tail

Appendix 3. Core functions of secr

These are the core functions of **secr** 2.9 – the ones that you are most likely to use. S3 methods are marked with an asterisk.

Function	Purpose
`AIC`*	model selection, model weights
`covariates`*	extract or replace covariates of traps, capthist or mask
`derived`	compute density from conditional likelihood models
`make.mask`	construct habitat mask (= mesh)
`plot`*	plot capthist, traps or mask
`read.capthist`	input captures and trap layout from Density format, one call
`predict`*	compute 'real' parameters for arbitrary levels of predictor variables
`predictDsurface`*	evaluate density surface at each point of a mask
`region.N`	compute expected and realised population size in specified region
`secr.fit`	maximum likelihood fit; result is a fitted 'secr' object
`summary`*	summarise capthist, traps or mask
`traps`*	extract or replace traps object in capthist

Appendix 4. Classified index to secr functions

Here is an index of **secr** functions classified by use (some minor functions are omitted). S3 methods are marked with an asterisk.

- Manipulate core objects
- Attributes of traps object
- Attributes of capthist object
- Data for each detection
- Operate on fitted model(s)
- Mask diagnostics
- Specialised graphics
- Convert or export data
- Miscellaneous

Manipulate data objects	
`addCovariates`	add spatial covariates to 'traps' or 'mask'
`deleteMaskPoints`	edit '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'
`randomHabitat`	generates habitat mask with random landscape
`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 detectors or occasions; change detector type
`sim.capthist`	simulate capture histories
`snip`	split transect(s) into equal sections
`subset`*	filter 'capthist', 'traps' or 'mask'
`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
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
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)
`signalmatrix`	sound x microphone table
`telemetryxy`	coordinates of telemetry fixes
`traps`*	embedded traps object(s)
Data for each detection	
`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(s)	
`ip.secr`	fit simple SECR model by simulation | inverse prediction
`par.secr.fit`	parallel secr.fit() (several models, using multiple cores)
`secr.fit`	maximum likelihood fit; result is a fitted `secr` object
Operate on fitted model(s)	
`AIC`*	model selection, model weights
`coef`*	'beta' parameters
`collate`	tabulate estimates from several models
`confint`*	profile likelihood confidence intervals
`CVa, CVa0`	CV of individual detection from fitted mixture model
`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

`ellipse.secr`	confidence ellipses for estimated parameters
`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 AIC or AICc weights
`par.derived`	parallel derived()
`par.region.N`	parallel region.N()
`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
`region.N`	expected and realised population size in specified region
`score.test`	model selection with score statistic using observed information
`secr.test`	Monte Carlo goodness-of-fit tests
`simulate`*	generate realisations of fitted model
`sim.secr`	parametric bootstrap
`vcov`*	variance-covariance matrix of 'beta' or 'real' parameters
Mask diagnostics	
`esa.plot`	cumulative plot esa vs buffer width
`mask.check`	likelihood or estimates vs. buffer width and spacing
`suggest.buffer`	find buffer width to keep bias within bounds
Specialised graphics	
`buffer.contour`	concave and convex boundary strips
`fx.total`	summed pdfs of home-range centre pdfs (use with `plot.Dsurface`)
`fxi.contour`	contour plot of home-range centre pdf(s)
`pdot.contour`	contour plot of detection probability
`strip.legend`	add colour legend to existing plot
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`
`clone`	replicate points to emulate overdispersion
`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
`CV`	coefficient of variation
`dbar`	mean distance between capture locations
`distancetotrap`	from an arbitrary set of points
`edist`	Euclidean distance
`MMDM`	mean maximum distance moved
`moves`	distances between capture locations
`nearesttrap`	from an arbitrary set of points
`nedist`	Non-Euclidean distance
`pdot`	location-specific net probability of detection
`PG`	proportion of telemetry fixes in given polygons
`RPSV`	'root pooled spatial variance', a simple measure of home-range size

Appendix 5. Datasets

See each help page for details e.g., ?deermouse

`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)

`OVpossum`

Brushtail possum *Trichosurus vulpecula* live trapping in the Orongorongo Valley, Wellington, New Zealand 1996–1997 (Efford and Cowan In: *The Biology of Australian Possums and Gliders* Goldingay and Jackson eds. Pp. 471–483).

`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 6. More on models in secr

A family of capture–recapture models, such as the Cormack-Jolly-Seber models for survival, may include submodels[5] 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. Set link functions with the 'link' argument of `secr.fit`.

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. Three real parameters are commonly modelled in **secr**; these are denoted D (for density), g0, and sigma. Only the last two 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.).

Other 'real' parameters appear in particular contexts. '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`). 'lambda0' substitutes for 'g0' when the detection function is defined in terms of cumulative hazard. 'pmix' represents the mixing proportion in finite mixture models (or e.g., the sex ratio in hybrid mixture models with 'hcov').

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 β 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 β 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 modelling in R. $\mathbf{X}$ has one column for each element of β. 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 2014). The necessary design data are either extracted from the inputs or generated automatically, as explained in later sections. 'Real' parameters fall into two groups:

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

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 2014) 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.