secr 5.6 - spatially explicit capture–recapture in R

Murray Efford

2026-11-25

Contents

Introduction to SECR **1**
- State and observation models . . . 2
- Distribution of home-range centres . . . 2
- Detection functions . . . 3
- Detector types . . . 3

Outline of the package secr **4**
- How **secr** works . . . 4
- Model fitting and estimation . . . 5
- Habitat masks . . . 6
- Input . . . 6
- Output . . . 6

Documentation **6**

Acknowledgment **7**

References **7**

Appendix 1. Core functions of secr **9**

Appendix 2. Classified index to secr functions **10**

This document provides a compact overview of **secr** 5.6, an R package for spatially explicit capture–recapture analysis (SECR). For background on SECR and a tutorial see the SECR book, Efford (2025).

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).
- **ipsecr** fits models by simulation and inverse prediction, rather than maximum likelihood; this is a rigorous way to analyse data from single-catch traps (ipsecr-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).
- **openCR** implements spatial open-population capture–recapture models (Efford and Schofield 2020) (openCR-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 in a region defined by the researcher.

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 areas (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, Turek et al. 2021); this approach is slower than ML SECR and 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. 1a). 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.

Detection functions

A detection model describes the decline in detection probability with distance (d) from the home-range centre (Fig. 1b). 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).

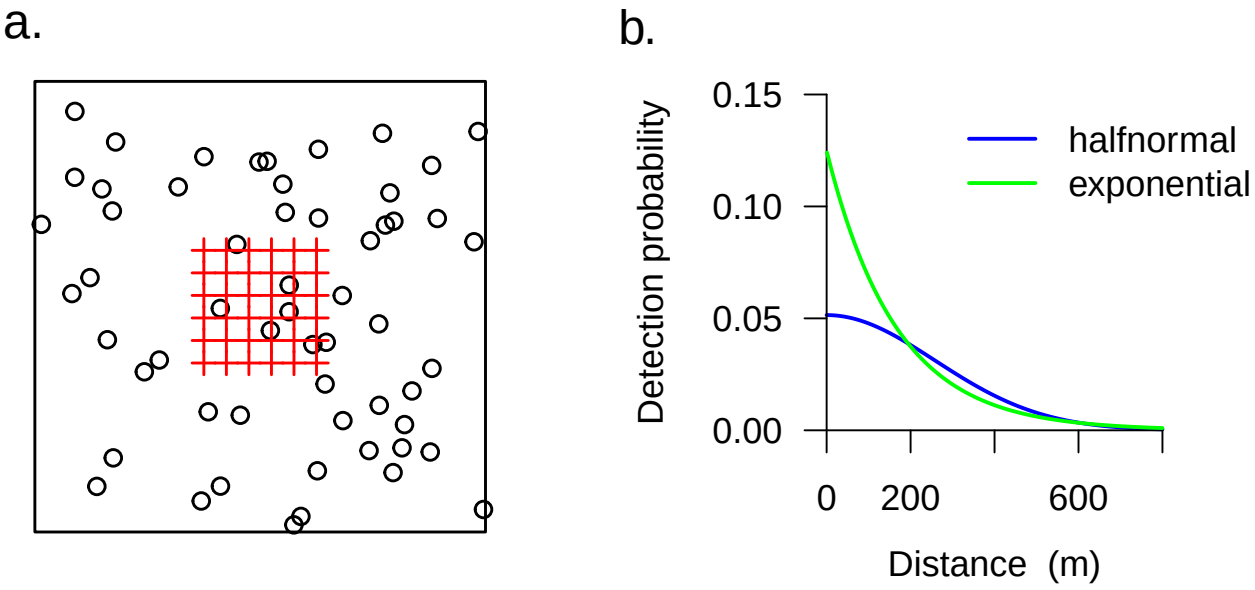

Fig. 1. (a) Hypothetical Poisson distribution of range centres near an array of detectors. Each dot represents one individual. SECR estimates the density of this distribution. (b) Alternative detection functions. The halfnormal is defined by $g(d) = g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right)$ and the exponential by $g(d) = g_0 \exp\left(-\frac{d}{\sigma}\right)$. See ?detectfn for more.

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. Estimates of average density using the multi-catch likelihood for single-catch data appear only slightly biased (Efford, Borchers and Byrom 2009), and this substitution is made automatically in **secr**, with a warning. However, the substitution is imperfect when density varies

(Distiller and Borchers 2015). Simulation and inverse prediction in **ipsecr** is an alternative and more robust method for single-catch data.

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. 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 in **secr** for the analysis of locational data from telemetry ('telemetry' detector type). Telemetry data are used to augment capture–recapture data (see `addTelemetry`). A sample of radio-collared animals can also be used to calibrate counts of the cues of unmarked animals (telemetry-calibrated cue counts; see `make.marking`).

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

Outline of the package secr

The program DENSITY (Efford et al. 2004, Efford 2012) provides a graphical interface to SECR methods that was used 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 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** 5.6 uses external C++ code via package **Rcpp** for computationally intensive operations (Eddelbuettel and Francois 2011); Multi-threading on multiple CPUs with **RcppParallel** (Allaire et al. 2021) gives major speed gains. The most important functions of **secr** are listed in Appendix 1.

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**.

[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.

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. 2 indicates the relationships among the classes.

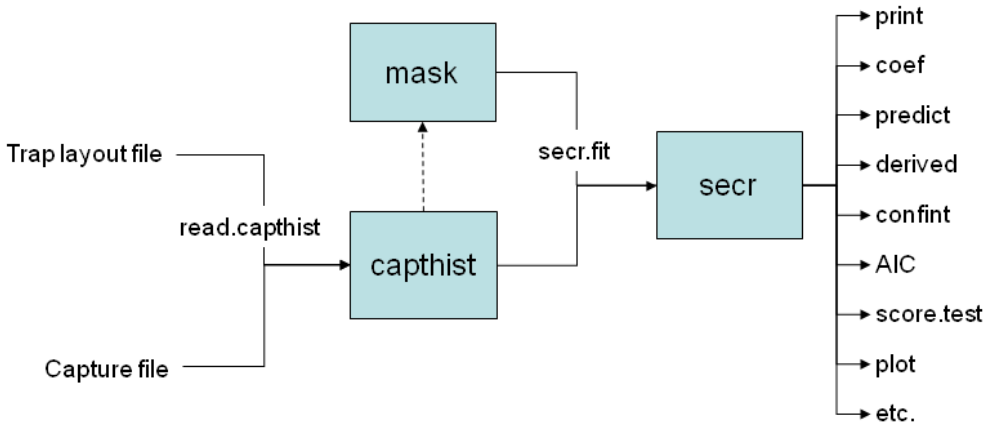

Fig. 2. Essentials of the **secr** package.

- 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.
- 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. Use `addCovariates` for covariates from spatial data sources (e.g., shapefile or 'sf' object)
- Fitted secr models may be manipulated with the methods shown on the right and others listed in Appendix 2.

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

[2]Text in this font refers to R objects that are documented in online help for the **secr** package, or in base R.

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 `sim.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; 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 Efford (2025) and the help for 'mask' and the various mask-related functions (`make.mask`, `mask.check`, `suggest.buffer`, and `esaPlot`). 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.

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 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 2.

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 in the file secr-manual.pdf. Searching this text is a powerful way to locate a function for a particular task.

The web page https://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. For information on changes in each version, type at the R prompt:

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

Help may be sought on the Density | secr forum at www.phidot.org. Another forum intended for both software issues and wider discussion is secrgroup.

Acknowledgment

Sonnet 5.5 (Anthropic) was used through Claude Code, to write and test parts of **secr** 5.6, especially the code for telemetry-calibrated cue counts (telemetrytype 'marking').

References

Allaire, J. J., Francois, R., Ushey, K., Vandenbrouck, G., Geelnard, M. and Intel (2021) RcppParallel: Parallel Programming Tools for 'Rcpp'. R package version 5.1.4. https://CRAN.R-project.org/package=RcppParallel.

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.

Distiller, G. and Borchers, D. L. (2015) A spatially explicit capture–recapture estimator for single-catch traps. *Ecology and Evolution* **5**, 5075–5087.

Eddelbuettel, D. and Francois, R. (2011) Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software* **40(8)**, 1–18. https://www.jstatsoft.org/v40/i08/.

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 https://www.otago.ac.nz/density/.

Efford, M. G. (2023) ipsecr: An R package for awkward spatial capture–recapture data. *Methods in Ecology and Evolution* **14**, 1182–1189.

Efford, M. G. (2025) The SECR book. A handbook of spatially explicit capture–recapture methods. Version 1.0. Zenodo https://doi.org/10.5281/zenodo.15109937 and https://murrayefford.github.io/SECRbook/.

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. 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.

Efford, M. G. and Schofield, M. R. (2020) A spatial open-population capture–recapture model. *Biometrics* **76**, 392–402.

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., 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.

Turek, D., Milleret, C., Ergon, T., Brøseth, H., Dupont, P., Bischof, R., and Valpine, P. de. (2021) Efficient estimation of large-scale spatial capture–recapture models.*Ecosphere* **12(2)** https://doi.org/10.1002/ecs2.3385

Appendix 1. Core functions of secr

These are the core functions of **secr** 5.6 – the ones that you are most likely to use. S3 methods are marked with an asterisk.

Function	Purpose
`addCovariates`	add spatial covariates to traps or mask
`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 2. 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

Function	Purpose
Manipulate data objects	
`addCovariates`	add spatial covariates to 'traps' or 'mask'
`append.capthist`	combine independent 'capthist' objects
`as.mask`	coerce 'traps' object to 'mask' for specialised plotting
`binCovariate`	bin a continuous covariate
`deleteMaskPoints`	edit 'mask'
`discretize`	rasterize area-search capthist data
`dropTelemetry`	remove telemetry occasions and notional detector from 'capthist'
`fillNA`	replace missing covariate values
`head`*	first rows of 'capthist', 'traps' or 'mask'
`join`	combine sessions of multi-session 'capthist' object
`make.grid`	construct detector array
`make.circle`	construct circular detector array
`make.poly`	construct polygon detector array
`make.transect`	construct transect detector array
`make.capthist`	form 'capthist' from 'traps' and detection data
`make.mask`	construct habitat mask (mesh)
`make.systematic`	construct clustered random systematic design
`make.lacework`	construct lacework random systematic design
`make.spcosa`	construct spatial coverage design
`MS.capthist`	combine 'capthist' objects into one multi-session 'capthist'
`plot`*	plot 'capthist', 'traps' or 'mask'
`plotMaskEdge`	draw line around mask cells
`randomHabitat`	generates habitat mask with random landscape
`rbind`*	append 'capthist', 'traps', 'popn' or 'mask' objects
`read.capthist`	input captures and trap layout from Density format, one call
`read.mask`	input habitat mask from file
`read.telemetry`	input telemetry fixes from file or dataframe
`read.traps`	input detector locations from text file
`reduce`*	aggregate detectors or occasions; change detector type
`shareFactorLevels`	give factor covariates the same levels across sessions
`sim.capthist`	simulate capture histories
`sim.popn`	simulate 2-D population
`sim.resight`	simulate mark-resight data
`simOU`	simulate Ornstein-Uhlenbeck movement
`simOU.capthist`	simulate detections with Ornstein-Uhlenbeck movement
`sightingPlot`	bubble plot of sightings in capthist object

Function	Purpose
`snip`	split transect(s) into equal sections
`split`*	split a single-session capthist or mask by various criteria
`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.)
`markocc`	vector distinguishing marking and sighting occasions
`polyID`*	polygon or transect identifier
`telemetrytype`*	telemetry type ('concurrent', 'dependent', 'independent', 'marking')
`timevaryingcov`	name time-varying covariate(s)
`usage`*	occasion- and detector-specific effort
Attributes of capthist object	
`addSightings`	add sighting data to a 'proximity', 'count' or 'polygon' object
`addTelemetry`	add telemetry data to a 'proximity' or 'count' object
`covariates`*	individual-level covariates, including grouping factors
`make.marking`	capthist for telemetry-calibrated cue counts
`marked`	when telemetry collars working
`session`*	session identifier(s)
`signalmatrix`	sound x microphone table
`telemetryxy`	coordinates of telemetry fixes
`Tm`	counts of marked animals that were not identified
`traps`*	embedded traps object(s)
`Tu`	counts of unmarked animals
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)	
`list.secr.fit`	fit several models and return secrlist object
`secr.fit`	maximum likelihood fit; result is a fitted `secr` object
`secr.refit`	repeat or resume a model fit
Operate on fitted model(s)	
`adjustVarD`	apply c-hat to density SE and confidence intervals
`AIC`*	model selection, model weights

Function	Purpose
`AICcompatible`*	whether models can be compared by AIC
`chat.nj`	overdispersion of activity centres
`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
`derivednj`	variance from replicated sampling units
`derivedCluster`	variance from replicated sampling units
`derivedDcoef`	derive intercept and other coefficients from relative density model
`derivedDfit`	derive absolute density model from relative density model
`derivedDsurface`	derive absolute density surface from relative density model
`derivedExternal`	variance from replicated sampling units
`ellipse.secr`	confidence ellipses for estimated parameters
`esa`*	effective sampling area by individual
`fxi`*	probability density of home-range centre
`LLsurface`*	compute likelihood surface and plot contours
`logLik`*	log-likelihood of fitted model
`LR.test`	likelihood-ratio test of two models
`makeNewData`*	default design data for `predict`
`MCgof`*	Monte Carlo goodness-of-fit after Choo et al. 2024
`modelAverage`*	combine estimates using AIC or AICc weights
`plot`*	plot detection functions with confidence bands
`predict`*	'real' parameters for arbitrary levels of predictor variables
`predictDlambda`	density trend from a Dlambda model
`predictDsurface`*	evaluate density surface at each point of a mask
`region.N`*	expected and realised population size in specified region
`RSE`	extract precision (relative SE) of 'real' parameter estimates
`score.test`	model selection with score statistic using observed information
`secr.test`	Monte Carlo goodness-of-fit tests
`secrlist`	combine fitted models as a 'secrlist' object
`simulate`*	generate realisations of fitted model
`sim.secr`	parametric bootstrap
`vcov`*	variance-covariance matrix of 'beta' or 'real' parameters
Mask diagnostics	
`esaPlot`	cumulative plot esa or $\hat{D}$ 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	
`bufferContour`	concave and convex boundary strips
`fxTotal`	summed pdfs of home-range centre pdfs (use with `plot.Dsurface`)
`fxiContour`	contour plot of home-range centre pdf(s)
`pdotContour`	contour plot of detection probability
`strip.legend`	add colour legend to existing plot
`usagePlot`	plot usage, detections or sightings
Convert or export data	
`rast`	convert 'mask' or 'Dsurface' to a **terra** SpatRaster

Function	Purpose
`raster`	convert 'mask' or 'Dsurface' to a **raster** RasterLayer
`RMarkInput`	convert 'capthist' to dataframe for RMark
`write.capthist`	export 'capthist' as text files for DENSITY
`write.captures`	write detections as text (lower-level than `write.capthist`)
`write.DA`	convert 'capthist' for analysis in WinBUGS
`write.mask`	write mask as text
`write.traps`	write detector locations as text
`writeGPS`	upload coordinates to GPS using GPSBabel
Miscellaneous	
`ARL`	asymptotic range length
`boundarytoSF`	convert polygon formats (matrix, **sp**, **sf**, **terra**) to 'sfc'
`as.popn`	coerce **spatstat** ppp object to 'popn'
`autoini`	generate starting values of D, g0 and sigma for `secr.fit`
`centroids`	centroid of each animal's detections
`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
`Enk`	expected individuals per detector
`flip`*	reflect a point array
`gridCells`	grid cells around points (e.g. traps object)
`kfn`	overlap index for halfnormal home range
`maskarea`	area of a mask
`MMDM`	mean maximum distance moved
`moves`	distances between capture locations
`nearesttrap`	from an arbitrary set of points
`nedist`	Non-Euclidean distance
`ORL`	observed range length
`pdot`	location-specific net probability of detection
`PG`	proportion of telemetry fixes in given polygons
`pmixProfileLL`	profile likelihood as function of mixing proportion
`rotate`*	rotate a point array
`RPSV`	'root pooled spatial variance', a simple measure of home-range size
`setNumThreads`	sets environment variable RCPP_PARALLEL_NUM_THREADS
`shift`*	translate a point array
`spacing`*	detector or mask spacing
`t2r2`	serial correlation of detection locations
`trapsPerAnimal`	frequency distribution of detectors per animal