Package 'secr'

March 11, 2010

Type Package

Title Spatially explicit capture-recapture

Version 1.3.0

Depends R (>= 2.10.0), abind, MASS, nlme, stats

Date 2010-03-11

Author Murray Efford

Maintainer Murray Efford <murray.efford@otago.ac.nz>

Description Estimate animal population density with capture–recapture data from an array of passive dectors (traps). Models incorporating distance-dependent detection are fitted by maximizing the likelihood. Tools are included for data manipulation and model selection.

License GPL (>=2)

ZipData yes

URL http://www.otago.ac.nz/density

R topics documented:

`secr-package` *Spatially Explicit Capture–Recapture Models*

Description

Analyse data from a spatially distributed animal population sampled with an array of passive detectors, such as traps.

Details

Package:	secr
Type:	Package
Version:	1.3.0
Date:	2010-03-11
License:	GNU General Public License Version 2 or later

Warning: Version 1.3.0 should be viewed as a beta release: some functions may not work with all documented settings. Feedback is very welcome, including suggestions for additional documentation or new features consistent with the overall design.

Data comprise the locations of detectors (traps) in an object of class 'traps' and the detection histories of individually marked animals in an object of class 'capthist'. Models for population density and detection are defined using symbolic formula notation. Possible predictors for detection probability include several pre-defined variables (t, b etc.) corresponding to 'time', 'behaviour' and other effects. Habitat is distinguished from nonhabitat with an object of class 'mask'. Models are fitted by maximizing either the full likelihood or the likelihood conditional on the number of individuals (n). Conditional likelihood models, while limited to homogeneous Poisson density, allow continuous individual covariates for detection. Fitting creates an object of class `secr`. Generic methods (plot, print, summary etc.) are available for each object class.

A more extensive overview can be got by typing

RShowDoc ('secr-overview', package='secr')

at the R prompt after the package has been loaded.

The analyses in **secr** extend those available in the software Density (see www.otago.ac.nz/density for the most recent version of Density).

Acknowledgements

David Borchers made these methods possible with his work on the likelihood, and I'm grateful for his continuing advice. Jeff Laake provided encouragement and reviewed an early version. Ray Brownrigg got my Windows code running under Unix. Deanna Dawson editted some of the documentation (the cleaner bits!) and her support and collaboration were important throughout.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

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

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

Efford, M. G., Dawson, D. K. and Robbins C. S. (2004) DENSITY: software for analysing capture-recapture data from passive detector arrays. *Animal Biodiversity and Conservation* **27**, 217–228.

See Also

secr.fit, traps, capthist, mask

Examples

```
## Not run:

## generate some data & plot
detectors  <- make.grid (nx = 10, ny = 10, spacing = 20,
    detector = 'multi')
plot(detectors, label = TRUE, border = 0, gridspace = 20)
detections <- sim.capthist (detectors, noccasions = 5,
    popn = list(D = 5, buffer = 100),
    detectpar = list(g0 = 0.2, sigma = 25))
session(detections) <- 'Simulated data'
plot(detections, border = 20, tracks = TRUE, varycol = TRUE)

## generate habitat mask
mask <- make.mask (detectors, buffer = 100, nx = 48)

## fit model and display results
secr.model <- secr.fit (detections, model = g0~b, mask = mask)
secr.model


## End(Not run)
```

AIC.secr *Compare SECR Models*

Description

Terse report on the fit of one or more spatially explicit capture–recapture models. Models with smaller values of AIC (Akaike's Information Criterion) are preferred.

Usage

```
## S3 method for class 'secr':
AIC(object, ..., sort = TRUE, k = 2, dmax = 10)
```

Arguments

object	secr object output from the function secr.fit
...	other secr objects
sort	logical for whether rows should be sorted by ascending AICc
k	numeric, the penalty per parameter to be used; always k = 2 in this method
dmax	numeric, the maximum AIC difference for inclusion in confidence set

Details

Models to be compared must have been fitted to the same data and use the same likelihood method (full vs conditional).

AIC with small sample adjustment is given by

$$\text{AIC}_c = -2\log(L(\hat{\theta})) + 2K + \frac{2K(K+1)}{n-K-1}$$

where K is the number of 'beta' parameters estimated. The sample size n is the number of individuals observed at least once (i.e. the number of rows in capthist).

Model weights are calculated as

$$w_i = \frac{\exp(-\Delta_i/2)}{\sum \exp(-\Delta_i/2)}$$

Models for which dAICc > dmax are given a weight of zero and are excluded from the summation. Model weights may be used to form model-averaged estimates of real or beta parameters with model.average (see also Buckland et al. 1997, Burnham and Anderson 2002).

The argument k is included for consistency with the generic method AIC.

Value

A data frame with one row per model. By default, rows are sorted by ascending AICc.

model	character string describing the fitted model
detectfn	shape of detection function fitted (halfnormal vs hazard-rate)
npar	number of parameters estimated
logLik	maximized log likelihood

AIC	Akaike's Information Criterion
AICc	AIC with small-sample adjustment of Hurvich & Tsai (1989)
dAICc	difference between AICc of this model and the one with smallest AICc
AICwt	AICc model weight

Note

The issue of goodness-of-fit and possible adjustment of AIC for overdispersion has yet to be addressed (cf QAIC in MARK).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Buckland S. T., Burnham K. P. and Augustin, N. H. (1997) Model selection: an integral part of inference. *Biometrics* **53**, 603–618.

Burnham, K. P. and Anderson, D. R. (2002) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Second edition. New York: Springer-Verlag.

Hurvich, C. M. and Tsai, C. L. (1989) Regression and time series model selection in small samples. *Biometrika* **76**, 297–307.

See Also

model.average, AIC, secr.fit, print.secr, score.test, LR.test, deviance.secr

Examples

```
## Compare two models fitted previously
## secrdemo.0 is a null model
## secrdemo.b has a learned trap response

data(secrdemo)
AIC(secrdemo.0, secrdemo.b)
```

---

autoini *Initial Parameter Values for SECR*

Description

Find plausible initial parameter values for secr.fit. A simplified model is fitted by a fast ad hoc method.

Usage

```
autoini(capthist, mask, detectfn = 0, thin = 0.2)
```

Arguments

capthist	capthist object
mask	mask object compatible with the detector layout in capthist
detectfn	shape of detection function 0 = half-normal
thin	proportion of points to retain in mask

Details

Plausible starting values are needed to avoid numerical problems when fitting SECR models. Actual models to be fitted will usually have more than the three basic parameters output by autoini; other initial values can usually be set to zero for secr.fit. If the algorithm encounters problems obtaining a value for g0, the default value of 0.1 is returned.

Only the half-normal detection function is available in autoini (cf other options in e.g. sim.capthist).

autoini implements a modified version of the algorithm proposed by Efford et al. (2004). In outline, the algorithm is

1. Find value of sigma that predicts the 2-D dispersion of individual locations (see RPSV)
2. Find value of g0 that, with sigma, predicts the observed mean number of captures per individual (by algorithm of Efford et al. (2009, Appendix 2))
3. Compute the effective sampling area from g0, sigma, using thinned mask (see esa)
4. Compute D = n/esa(g0, sigma), where n is the number of individuals detected

Here 'find' means solve numerically for zero difference between the observed and predicted values, using uniroot.

If RPSV cannot be computed the algorithm tries to use observed mean recapture distance $\bar{d}$. Computation of $\bar{d}$ fails if there no recaptures, and all returned values are NA.

A proportion 1–thin of the points in the mask may be discarded at random to speed execution.

Value

A list of parameter values :

D	Density (animals per hectare)
g0	Magnitude (intercept) of detection function
sigma	Spatial scale of detection function (m)

Note

autoini may in future include an option to use RPSV instead of dbar.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G., Dawson, D. K. and Robbins C. S. (2004) DENSITY: software for analysing capture–recapture data from passive detector arrays. *Animal Biodiversity and Conservation* **27**, 217–228.

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.

See Also

capthist, mask, secr.fit, dbar

Examples

```
demotraps <- make.grid()
demomask <- make.mask(demotraps)
demoCH <- sim.capthist (demotraps, popn = list(D = 5, buffer = 100))
autoini (demoCH, demomask)
```

---

captdata *Detection Dataset*

Description

An object of class capthist containing the demonstration data from program Density 4.1 (Efford 2007), specifically the trap layout data in 'trap.txt' and the captures in 'capt.txt'.

Usage

```
data(captdata)
```

Details

These data are entirely fictitious. They comprise 235 captures of 76 animals in 100 single-catch traps over 5 occasions.

Source

Efford, M.G. (2007) Density 4.1: software for spatially explicit capture-recapture. Department of Zoology, University of Otago, Dunedin, New Zealand. http://www.otago.ac.nz/density.

Examples

```
data(captdata)
plot(captdata)
secr.fit(captdata)
```

---

capthist *Spatial Capture History Object*

Description

A capthist object encapsulates all data needed by secr.fit, except for the optional habitat mask.

Details

An object of class `capthist` holds spatial capture histories, detector (trap) locations, individual covariates and other data needed for a spatially explicit capture-recapture analysis with `secr.fit`.

For 'single' and 'multi' detectors, `capthist` is a matrix with one row per animal and one column per occasion (i.e. dim(capthist) = c(nc, noccasions)); each element is either zero (no detection) or a detector number. For other detectors ('proximity', 'count', 'signal' etc.), `capthist` is an array of values and dim(capthist) = c(nc, noccasions, ntraps); values maybe binary ({–1, 0, 1}), integer or real depending on the detector type.

Deaths during the experiment are represented as negative values.

Ancillary data are retained as attributes of a `capthist` object as follows:

- traps – object of class `traps` (required)
- session – session identifier (required)
- covariates – dataframe of individual covariates (optional)
- cutval – threshold of signal strength for detection ('signal' only)
- signal – signal strength values, one per detection ('signal' only)
- detectedXY – dataframe of coordinates for location within polygon ('polygon' only)

The parts of a capthist object can be assembled with the function `make.capthist`. Use `sim.capthist` for Monte Carlo simulation (simple models only). Methods are provided to display and manipulate `capthist` objects (print, summary, plot, rbind, subset, reduce) and to extract and replace attributes (covariates, traps, xy).

A multi-session `capthist` object is a list in which each component is a `capthist` for a single session. The list maybe derived directly from multi-session input in Density format, or by combining existing `capthist` objects with `MS.capthist`.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

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

See Also

`traps`, `secr.fit`, `make.capthist`, `sim.capthist`, `subset.capthist`, `rbind.capthist`, `MS.capthist`, `reduce.capthist`, `mask`

`capthist.parts` *Dissect Spatial Capture History Object*

Description

Extract parts of an object of class 'capthist'.

Usage

```
animalID(object, names = TRUE)
occasion(object)
trap(object, names = TRUE)
xy(object)
signal(object)
xy(object) <- value
signal(object) <- value
```

Arguments

`object`	a 'capthist' object
`names`	if FALSE the values returned are numeric indices rather than names
`value`	replacement value (see Details)

Details

These functions extract data on detections, ignoring occasions when an animal was not detected.

`trap` returns polygon or transect numbers if `traps(object)` has detector type 'polygon' or 'transect'.

Replacement values must precisely match `object` in number of detections in their order. `xy<-` expects a dataframe of x and y coordinates for points of detection within a 'polygon' or 'transect' detector.

Value

For `animalID` and `trap` a vector of numeric or character values, one per detection.

For `occasion`, a vector of numeric values, one per detection.

For `xy`, a dataframe with one row per detection and columns 'x' and 'y'.

For `signal`, a numeric vector with one element per detection.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`capthist`, `polyID`

Examples

```
data(captdata)
animalID(captdata)

temp <- sim.capthist(popn=list(D=1), make.grid(detector='count'))
cbind(ID=as.numeric(animalID(temp)), occ=occasion(temp), trap=trap(temp))
```

---

coef.secr *Coefficients of secr Object*

Description

Extract coefficients (estimated beta parameters) from a spatially explicit capture–recapture model.

Usage

```
## S3 method for class 'secr':
coef (object, alpha = 0.05, ...)
```

Arguments

object	secr object output from secr.fit
alpha	alpha level
...	other arguments (not used currently)

Value

A data frame with one row per beta parameter and columns for the coefficient, SE(coefficient), asymptotic lower and upper 100(1–alpha) confidence limits.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

secr.fit

Examples

```
## load & extract coefficients of previously fitted null model
data(secrdemo)
coef(secrdemo.0)
```

`confint.secr` *Profile Likelihood Confidence Intervals*

Description

Compute profile likelihood confidence intervals for 'beta' or 'real' parameters of a spatially explicit capture-recapture model,

Usage

```
## S3 method for class 'secr':
confint (object, parm, level = 0.95, newdata = NULL,
tracelevel = 1, tol = 0.0001, ...)
```

Arguments

`object`	`secr` model object
`parm`	numeric or character vector of parameters
`level`	confidence level (1 – alpha)
`newdata`	optional dataframe of values at which to evaluate model
`tracelevel`	integer for level of detail in reporting (0,1,2)
`tol`	absolute tolerance (passed to uniroot)
`...`	other arguments (not used)

Details

If `parm` is numeric its elements are interpreted as the indices of 'beta' parameters; character values are interpreted as 'real' parameters. Different methods are used for beta parameters and real parameters. Limits for the j-th beta parameter are found by a numerical search for the value satisfying $-2(l_j(\beta_j) - l) = q$, where l is the maximized log likelihood, $l_j(\beta_j)$ is the maximized profile log likelihood with β_j fixed, and q is the $100(1-\alpha)$ quantile of the χ^2 distribution with one degree of freedom. Limits for real parameters use the method of Lagrange multipliers (Fletcher and Faddy 2007), except that limits for constant real parameters are backtransformed from the limits for the relevant beta parameter.

Value

A matrix with one row for each parameter in `parm`, and columns giving the lower (lcl) and upper (ucl) 100*`level`

Note

Calculation may take a long time, so probably you will do it only after selecting a final model.

The R function `uniroot` is used to search for the roots of $-2(l_j(\beta_j) - l) = q$ within a suitable interval. The interval is anchored at one end by the MLE, and at the other end by the MLE inflated by a small multiple of the asymptotic standard error (1, 2, 4 or 8 SE are tried in turn, using the smallest for which the interval includes a valid solution).

A more efficient algorithm was proposed by Venzon and Moolgavkar (1988); it has yet to be implemented in **secr**, but see `plkhci` in the package **Bhat** for another R implementation.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Evans, M. A., Kim, H.-M. and O'Brien, T. E. (1996) An application of profile-likelihood based confidence interval to capture–recapture estimators. *Journal of Agricultural, Biological and Experimental Statistics* **1**, 131–140.

Fletcher, D. and Faddy, M. (2007) Confidence intervals for expected abundance of rare species. *Journal of Agricultural, Biological and Experimental Statistics* **12**, 315–324.

Venzon, D. J. and Moolgavkar, S. H. (1988) A method for computing profile-likelihood-based confidence intervals. *Applied Statistics* **37**, 87–94.

Examples

```
## Not run:
data (secrdemo)
## Limits for the constant real parameter 'D'
confint(secrdemo.0, 'D')

## End(Not run)
```

---

covariates *Covariates Attribute*

Description

Extract or replace covariates

Usage

```
covariates(object, ...)
covariates(object) <- value
```

Arguments

object	an object of class traps, popn, capthist, or mask
value	a dataframe of covariates
...	other arguments (not used)

Details

For replacement, the number of rows of value must match exactly the number of rows in object.

Value

covariates(object) returns the dataframe of covariates associated with object. covariates(object) may be NULL.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
temptrap <- make.grid(nx = 6, ny = 8)
covariates (temptrap) <- data.frame(halfnhalf =
    factor(rep(c('left','right'),c(24,24))) )
summary(covariates(temptrap))
```

D.designdata *Construct Density Design Data*

Description

Internal function used by secr.fit, confint.secr, and score.test.

Usage

```
D.designdata (mask, Dmodel, grps, sessionlevels, sessioncov = NULL)
```

Arguments

mask	mask object.
Dmodel	formula for density model
grps	vector of group names
sessionlevels	vector of character values for session names
sessioncov	optional dataframe of values of session-specific covariate(s).

Details

This is an internal **secr** function that you are unlikely ever to use. Unlike secr.design.MS, this function does *not* call model.matrix.

Value

Dataframe with one row for each combination of mask point, group and session. The dataframe has an attribute 'dimD' that gives the relevant dimensions: attr(dframe, 'dimD') = c(nmask, ngrp, R), where nmask is the number of mask points, ngrp is the number of groups, and R is the number of sessions. Columns correspond to predictor variables in Dmodel.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

secr.design.MS

`derived` *Derived Parameters of Fitted SECR Model*

Description

Compute derived parameters of spatially explicit capture-recapture model. Density is a derived parameter when a model is fitted by maximizing the conditional likelihood. So also is the effective sampling area (in the sense of Borchers and Efford 2008).

Usage

```
derived(object, sessnum = NULL, groups = NULL, alpha = 0.05,
    se.esa = FALSE, se.D = TRUE, loginterval = TRUE,
    distribution = NULL)
esa(object, sessnum = 1, beta = NULL, real = NULL)
```

Arguments

`object`	`secr` object output from `secr.fit`
`sessnum`	index of session in object\$capthist for which output required
`groups`	indices defining group(s) (see Details)
`alpha`	alpha level for confidence intervals
`se.esa`	logical for whether to calculate SE(mean(esa))
`se.D`	logical for whether to calculate SE(D-hat)
`loginterval`	logical for whether to base interval on log(D)
`distribution`	character string for distribution of the number of individuals detected
`beta`	vector of fitted parameters on transformed (link) scale
`real`	vector of 'real' parameters

Details

The derived estimate of density is a Horvitz-Thompson-like estimate:

$$\hat{D} = \sum_{i=1}^{n} a_i(\hat{\theta})^{-1}$$

where $a_i(\hat{\theta})$ is the estimate of effective sampling area for animal i with detection parameter vector θ.

A non-null value of the argument `distribution` overrides the value in `object$details`. The sampling variance of $\hat{D}$ from `secr.fit` by default is spatially unconditional (`distribution` = 'Poisson'). For sampling variance conditional on the population of the habitat mask (and therefore dependent on the mask area), specify `distribution` = 'binomial'. The equation for the conditional variance includes a factor $(1 - a/A)$ that disappears in the unconditional (Poisson) variance (Borchers and Efford 2007). Thus the conditional variance is always less than the unconditional variance. The unconditional variance may in turn be an overestimate or (more likely) an underestimate if the true spatial variance is non-Poisson.

Derived parameters may be estimated for population subclasses (groups) defined by the user with the `groups` argument. Each named factor in `groups` should appear in the covariates dataframe of object\$capthist (or each of its components, in the case of a multi-session dataset).

The effective sampling area 'esa' reported by `derived` is equal to the mean of the $a_i(\hat{\theta})$.

A 100(1–alpha)% asymptotic confidence interval is reported for density. By default, this is asymmetric about the estimate because the variance is computed by backtransforming from the log scale. You may also choose a symmetric interval (variance obtained on natural scale).

`esa` is used by `derived` to compute individual-specific effective sampling areas:

$$a_i(\hat{\theta}) = \int_A p.(\mathbf{X}; \mathbf{z}_i, \hat{\theta})\ \mathrm{d}\mathbf{X}$$

where $p.(\mathbf{X})$ is the probability an individual at X is detected at least once and the $\mathbf{z}_i$ are optional individual covariates. Integration is over the area A of the habitat mask.

The vector of detection parameters for `esa` may be specified via `beta` or `real`, with the former taking precedence. If neither is provided then the fitted values in `object$fit$par` are used. Specifying `real` parameter values bypasses the various linear predictors. Strictly, the 'real' parameters are for a naive capture (animal not detected previously).

The computation of sampling variances is relatively slow and may be suppressed with `se.esa` and `se.D` as desired.

Value

Dataframe with one row for each derived parameter ('esa', 'D') and columns as below

estimate	estimate of derived parameter
SE.estimate	standard error of the estimate
lcl	lower 100(1–alpha)% confidence limit
ucl	upper 100(1–alpha)% confidence limit
varcomp1	variance due to variation in n (Huggins' s^2)
varcomp2	variance due to uncertainty in estimates of detection parameters

For a multi-session or multi-group analysis the value is a list with one component for each session and group.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Borchers, D. L. and Efford, M. G. (2007) Supplements to Biometrics paper. Available online at http://www.otago.ac.nz/density.

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

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

See Also

`predict.secr`, `print.secr`, `secr.fit`

Examples

```
## extract derived parameters from a model fitted previously
## by maximizing the conditional likelihood
data(secrdemo)
derived (secrdemo.CL)

## what happens when sampling variance is conditional on mask N?
derived(secrdemo.CL, distribution = 'binomial')

## fitted g0, sigma
esa(secrdemo.CL)
## force different g0, sigma
esa(secrdemo.CL, real = c(0.2, 25))
```

`detectfn` *Detection Functions*

Description

A detection function relates the probability of detection to the distance of a detector from a point. The reference point is usually thought of as an animal's home-range centre. In **secr** only simple 2- or 3-parameter functions are used. Each type of function is identified by a numeric code (see below).

Some functions are defined only for simulation: these either cannot be fitted by maximum likelihood (uniform) or have yet to be implemented (compound halfnormal).

Code	Name	Parameters	Function
0	halfnormal	g0, sigma	$g(d) = g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right)$
1	hazard-rate	g0, sigma, z	$g(d) = g_0[1 - \exp\{-(d/\sigma)^{-z}\}]$
2	exponential	g0, sigma	$g(d) = g_0 \exp\{-(d/\sigma)\}$
3	compound halfnormal	g0, sigma, z	$g(d) = 1 - (1 - g_0 \exp\left(\frac{-d^2}{2\sigma^2}\right)^z)$
4	uniform	g0, sigma	$g(d) = g_0, d <= \sigma; g(d) = 0, otherwise\}$
5	w-exponential	g0, sigma, w	$g(d) = g_0, d < w; g(d) = g_0 \exp\{-(`d - w)/\sigma), otherwise\}$
9	binary signal strength	b0, b1	$g(d) = F(b_0 + b_1 d))$
10	signal strength	beta0, beta1, sdS	$g(d) = F((c - (\beta_0 + \beta_1 d))/sdS)$
11	signal strength spherical	beta0, beta1, sdS	$g(d) = F((c - (\beta_0 + \beta_1 (d-1) - 10 * log(d^2)/log(10)))/sdS)$

For functions (9), (10) and (11), 'F' is the standard normal distribution function and 'c' is an arbitrary signal threshold. The two parameters of (9) are functions of the parameters of (10) and (11): $b_0 = (\beta_0 - c)/sdS$ and $b_1 = \beta_1/sdS$ (see Efford et al. 2009).

Function (11) includes an additional 'hard-wired' term for sound attenuation due to spherical spreading. Detection probability at distances less than 1 m is given by $g(d) = F((c - \beta_0)/sdS)$

The hazard-rate detection function was described by Hayes and Buckland (1983). The compound halfnormal detection function follows Efford and Dawson (2009). The signal strength and binary signal strength functions are from Efford et al. (2009).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G. and Dawson, D. K. (2009) Effect of distance-related heterogeneity on population size estimates from point counts. *Auk* **126**, 100–111.

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.

Hayes, R. J. and Buckland, S. T. (1983) Radial-distance models for the line-transect method. *Biometrics* **39**, 29–42.

See Also

detectfnplot, secr detection models

---

detector *Detector Type*

Description

Extract or replace the detector type.

Usage

```
detector(object, ...)
detector(object) <- value
```

Arguments

object	object with 'detector' attribute e.g. traps
value	character string for detector type
...	other arguments (not used)

Details

Valid detector types in version 1.3 are 'single', 'multi', 'proximity', 'count', 'quadratbinary', 'quadratcount' and 'signal'. The detector type is stored as an attribute of a traps object. Detector types are mostly described by Efford et al. (2009a,b).

Value

character string for detector type

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

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

See Also

traps

Examples

```
## Default detector type is 'multi'
temptrap <- make.grid(nx = 6, ny = 8)
detector(temptrap) <- 'proximity'
summary(temptrap)
```

---

deviance *Deviance of fitted secr model and residual degrees of freedom*

Description

Compute the deviance or residual degrees of freedom of a fitted secr model, treating multiple sessions and groups as independent. The likelihood of the saturated model depends on whether the 'conditional' or 'full' form was used, and on the distribution chosen for the number of individuals observed (Poisson or binomial).

Usage

```
## S3 method for class 'secr':
deviance(object, ...)
## S3 method for class 'secr':
df.residual(object, ...)
```

Arguments

object	secr object from secr.fit
...	other arguments (not used)

Details

The deviance is $-2log(\hat{L}) + 2log(L_{sat})$, where $\hat{L}$ is the value of the log-likelihood evaluated at its maximum, and L_{sat} is the log-likelihood of the saturated model, calculated thus:

Likelihood conditional on n -

$L_{sat} = \log(n!) + \sum_{\omega}[n_{\omega}\log(\frac{n_{\omega}}{n}) - \log(n_{\omega}!)]$

Full likelihood, Poisson n -

$L_{sat} = n\log(n) - n + \sum_{\omega}[n_{\omega}\log(\frac{n_{\omega}}{n}) - \log(n_{\omega}!)]$

Full likelihood, binomial n -

$L_{sat} = n\log(\frac{n}{N}) + (N-n)\log(\frac{N-n}{N}) + \log(\frac{N!}{(N-n)!}) + \sum_{\omega}[n_{\omega}\log(\frac{n_{\omega}}{n}) - \log(n_{\omega}!)]$

n is the number of individuals observed at least once, n_{ω} is the number of distinct histories, and N is the number in a chosen area A that we estimate by $\hat{N} = \hat{D}A$.

The residual degrees of freedom is the number of distinct detection histories minus the number of parameters estimated. The detection histories of two animals are always considered distinct if they belong to different groups.

When samples are (very) large the deviance is expected to be distributed as χ^2 with $n_{\omega} - p$ degrees of freedom when p parameters are estimated. In reality, simulation is needed to assess whether a given value of the deviance indicates a satisfactory fit, or to estimate the overdispersion parameter c. `sim.secr` is a convenient tool.

Value

The scalar numeric value of the deviance or the residual degress of freedom extracted from the fitted model.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

See Also

`secr.fit`, `sim.secr`

Examples

```
data(secrdemo)
deviance(secrdemo.0)
df.residual(secrdemo.0)
```

`distancetotrap` *Distance To Nearest Detector*

Description

Compute distance from each of a set of points to the nearest detector in an array, or return the sequence number of the detector nearest each point.

Usage

```
distancetotrap(X, traps)

nearesttrap(X, traps)
```

Arguments

X	coordinates
traps	traps object

Details

`distancetotrap` returns the distance from each point in X to the nearest detector in `traps`. It may be used to restrict the points on a habitat mask.

Value

`distancetotrap` returns a vector of distances (assumed to be in metres).

`nearesttrap` returns the index of the nearest trap.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`make.mask`

Examples

```
## restrict a habitat mask to points within 70 m of traps
## this is nearly equivalent to using make.mask with the
## 'trapbuffer' option
temptrap <- make.grid()
tempmask <- make.mask(temptrap)
d <- distancetotrap(tempmask, temptrap)
tempmask <- subset(tempmask, d < 70)
```

---

ellipse.secr *Confidence ellipse*

Description

Plot joint confidence ellipse for two parameters of secr model

Usage

```
ellipse.secr(object, par = c("g0", "sigma"), alpha = 0.05,
    npts = 100, plot = TRUE, linkscale = TRUE, add = FALSE,
    col = palette(), ...)
```

Arguments

object	secr object output from secr.fit
par	character vector of length two, the names of two 'beta' parameters
alpha	alpha level for confidence intervals
npts	number of points on perimeter of ellipse
plot	logical for whether ellipse should be plotted
linkscale	logical; if FALSE then coordinates will be backtransformed from the link scale
add	logical to add ellipse to an existing plot
col	vector of one or more plotting colours
...	arguments to pass to plot functions

Details

A confidence ellipse is calculated from the asymptotic variance-covariance matrix of the beta parameters (coefficients), and optionally plotted.

If linkscale == FALSE, the inverse of the appropriate link transformation is applied to the coordinates of the ellipse, causing it to deform.

If object is a list of secr models then one ellipse is constructed for each model. Colours are recycled as needed.

Value

A list containing the x and y coordinates is returned invisibly

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
data(secrdemo)
ellipse.secr(secrdemo.0)
```

FAQ *Frequently Asked Questions, And Others*

Description

A place for hints and miscellaneous advice.

How do I install secr?

Under Windows, it is simplest to install the package binary from the Rgui. Save the file secr_1.3.x.zip to a local folder and use the menu option "Packages | Install packages(s) from local zip files...".

For other systems (see below) you will need to install the source package secr_1.3.x.tar.gz.

Whatever your system, you also need to get the package **abind** (use Packages | Install package(s)... to download from CRAN). Other required packages (**MASS**, **nlme**, **stats**) should be available as part of your R installation.

Does secr work on operating systems other than Windows?

Yes, at least in theory. From version 1.2.8 you should be able to install the source package secr_1.x.x.tar.gz if you have the necessary tools (check the 'R Installation and Administration' manual or the 'R for Mac OS X FAQ' on CRAN for help on installing source packages). Thanks to Ray Brownrigg and Joanne Potts for vital assistance. I would be glad to hear of successful installations on non-Windows systems (particularly if anyone produces a binary for Mac OS).

How can I get help?

There are three general ways of displaying documentation from within R. Firstly, you can bring up help pages for particular functions from the command prompt. For example:

```
?  secr.fit
```

Secondly, help.search() lets you ask for a list of the help pages on a vague topic. From R version 2.8.0 you can use just ??. For example:

```
??  'linear models'
```

Thirdly, you can display various documents:

```
RShowDoc ('secr-manual', package='secr')
RShowDoc ('secr-overview', package='secr')
```

See below for more R tips.

How should I report a problem?

If you get really stuck or find something you think is a bug then please report the problem. For the moment, the correct address is <density.software@otago.ac.nz>.

When reporting problems please send an actual dataset (ideally, the simplest one that exhibits the problem). Use `save` to wrap several R objects together in one .RData file, e.g., `save('captdata', 'secrdemo.0', 'secrdemo.b', file = 'mydata.RData')`. Also, paste the output from `packageDescription("secr")` into the text of your message.

Needless to say, we cannot promise to solve all problems.

Why do I get different answers from secr and Density?

Strictly speaking, this should not happen if you have specified the same model and likelihood, although you may see a little variation due to the different maximization algorithms. Likelihoods (and estimates) may differ if you use different integration meshes (habitat masks), which can easily happen because the programs differ in how they set up the mesh. If you want to make a precise comparison, save the Density mesh to a file and read it into **secr**, or vice versa.

Extreme data, especially rare long-distance movements, may be handled differently by the two programs. The 'minprob' component of the 'details' argument of `secr.fit` sets a threshold of probability for capture histories (smaller values are all set to minprob), whereas Density has no explicit limit. In the current version the default minprob has been reduced from 1e-20 to 1e-50. If you find a discrepancy with Density it may be worth lowering minprob even further.

How can I speed up model fitting and model selection?

If you don't need to model variation in density over space or time then consider maximizing the conditional likelihood in `secr.fit` (CL = TRUE). This reduces the complexity of the optimization problem, especially where there are several sessions and you want session-specific density estimates (by default, `derived` returns a separate estimate for each session even if the detection parameters are constant across sessions).

Check the extent and spacing of the habitat mask that you are using. Execution time is roughly proportional to the number of mask points. Default settings can lead to very large masks for detector arrays that are elongated 'north-south' because the number of points in the east-west direction is fixed. Compare results with a much sparser mask (e.g., nx=32 instead of nx=64).

Do you really need to fit that complex model? Chasing down small decrements in AIC is so last-century. Remember that detection parameters are mostly nuisance parameters, and models with big differences in AIC may barely differ in their density estimates. This is a good topic for further research - we seem to need a 'focussed information criterion' (Claeskens and Hjort 2008) to discern the differences that matter.

Use `score.test` to compare nested models. At each stage this requires only the more simple model to have been fitted in full; further processing is required to obtain a numerical estimate of the gradient of the likelihood surface for the more complex model, but this is much faster than maximizing the likelihood.

Will secr soon include feature X?

Finite mixture models Possibly. Mixture models do seem desirable as an option to allow for unmodelled heterogeneity. However, mixtures add an extra level of complexity across the entire detection model structure, and an additional 'real' parameter (for the mixing proportion) so the decision is not trivial. A decision was made to release and evaluate **secr** 1.3 before adding even more complexity.

Inverse prediction Yes, from version 1.2.7. Fitting SECR models by simulation and inverse prediction is still the only rigorous method for dealing with data from single-catch traps. However, maximizing the likelihood as if the data came from multi-catch traps yields fairly good density estimates even with single-catch data (estimates of detection g0 are biased). Limited capability for simulation-based fitting is provided by the function `ip.secr`. It is not intended to extend `ip.secr` to include a wider range of models.

Things You Might Need To Know About R

The function `findFn` in package **sos** lets you search CRAN for R functions by matching text in their documentation.

There is now a vast amount of R advice available on the web. For the terminally frustrated, 'R inferno' by Patrick Burns is recommended (www.burns-stat.com/pages/Tutor/R_inferno.pdf). "If you are using R and you think you're in hell, this is a map for you".

Method functions for S3 classes cannot be listed in the usual way by typing the function name at the R prompt because they are 'hidden' in a namespace. Get around this with getAnywhere(). For example:

```
getAnywhere(print.secr)
```

R objects have 'attributes' that usually are kept out of sight. Important attributes are 'class' (all objects), 'dim' (matrices and arrays) and 'names' (lists). **secr** hides quite a lot of useful data as named 'attributes'. Usually you will use summary and extraction methods (`traps`, `covariates`, `usage` etc.) to view and change the attributes of the various classes of object in **secr**. If you're curious, you can reveal the lot with 'attributes'. For example:

```
data(captdata)
traps(captdata) ## extraction method for 'traps'
attributes(captdata) ## all attributes
```

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Claeskens, G. and Hjort N. L. (2008) *Model Selection and Model Averaging*. Cambridge: Cambridge University Press.

`flip` *Flip Points*

Description

Flip an array of points about a vertical or horizontal axis.

Usage

```
flip (object, lr = F, tb = F, ...)
```

Arguments

`object`	a 2-column matrix or object that can be coerced to a matrix
`lr`	either logical for whether array should be flipped left-right, or numeric value for x-coordinate of axis about which it should be flipped left-right
`tb`	either logical for whether array should be flipped top-bottom, or numeric value for y-coordinate of axis about which it should be flipped top-bottom
`...`	other arguments (not used)

Details

Logical values for `lr` or `tb` indicate that points should be flipped about the mean on the relevant axis.

Numeric values indicate the particular axis value(s) about which points should be flipped. The default arguments result in no change.

This is a generic function. A method is provided for `traps` objects.

Value

A matrix with the coordinates of each point reflected about the desired axis or axes.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`, `rotate.traps`

Examples

```
temp <- matrix(runif (20) * 2 - 1, nc = 2)
temp2 <- flip(temp, lr = 1)
plot(temp, xlim=c(-1.5,4), ylim = c(-1.5,1.5), pch = 16)
points (temp2, pch = 1)
arrows (temp[,1], temp[,2], temp2[,1], temp2[,2], length = 0.1)
```

`flip.traps` *Flip Detector Array*

Description

Flip a detector array about a vertical or horizontal axis.

Usage

```
## S3 method for class 'traps':
flip (object, lr = F, tb = F, ...)
```

Arguments

`object`	a 2-column matrix or object that can be coerced to a matrix
`lr`	either logical for whether array should be flipped left-right, or numeric value for x-coordinate of axis about which it should be flipped left-right
`tb`	either logical for whether array should be flipped top-bottom, or numeric value for y-coordinate of axis about which it should be flipped top-bottom
`...`	other arguments (not used)

Details

Logical values for `lr` or `tb` indicate that points should be flipped about the mean on the relevant axis.

Numeric values indicate the particular axis value(s) about which points should be flipped.

The default arguments result in no change.

Value

Object of class `traps` with the coordinates of each point reflected about the desired axis or axes.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`, `rotate.traps`, `shift.traps`

Examples

```
par(mfrow=c(1,2), xpd = TRUE)
traps1 <- make.grid(nx = 8, ny = 6, ID = 'numxb')
traps2 <- flip (traps1, lr = TRUE)
plot(traps1, border = 5, lab = TRUE, offset = 7, gridl = FALSE)
plot(traps2, border = 5, lab = TRUE, offset = 7, gridl = FALSE)
```

`homerange` *Home Range Statistics*

Description

Two ad hoc measures of home range size may be calculated in **secr** from capture–recapture data:

`dbar` is the mean distance between consecutive capture locations, pooled over individuals (e.g. Efford 2004).

`RPSV` (for 'Root Pooled Spatial Variance') is a measure of the 2-D dispersion of the locations at which individual animals are detected, pooled over individuals.

Usage

```
dbar(capthist)
RPSV(capthist)
```

Arguments

`capthist`	object of class `capthist`

Details

`dbar` is defined as

$$\overline{d} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n_i-1}\sqrt{(x_{i,j}-x_{i,j+1})^2+(y_{i,j}-y_{i,j+1})^2}}{\sum_{i=1}^{n}(n_i-1)}$$

`RPSV` is defined as

$$RPSV = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n_i}[(x_{i,j}-\overline{x}_i)^2+(y_{i,j}-\overline{y}_i)^2]}{\sum_{i=1}^{n}(n_i-1)-1}}$$

`dbar` and `RPSV` have a specific role as proxies for detection scale in inverse-prediction estimation of density (Efford 2004; see `ip.secr`).

`dbar` is used in `autoini` to obtain plausible starting values for maximum likelihood estimation.

Value

Scalar distance in metres, or a list of such values if `capthist` is a multi-session list.

Note

Both measures are affected by the arrangement of detectors. `dbar` is also affected quite strongly by serial correlation in the sampled locations. Using `dbar` with 'proximity' detectors raises a problem of interpretation, as the original sequence of multiple detections within an occasion is unknown. RPSV is a value analogous to the standard deviation of locations about the home range centre.

Inclusion of these measures in the **secr** package does not mean they are recommended for general use! It is usually better to use a spatial parameter from a fitted model (e.g., σ of the half-normal

detection function). Even then, be careful that σ is not 'contaminated' with behavioural effects (e.g. attraction of animal to detector) or 'detection at a distance'.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

See Also

autoini

Examples

```
data(captdata)
dbar(captdata)
RPSV(captdata)
```

---

ip.secr *Spatially Explicit Capture–Recapture by Inverse Prediction*

Description

Estimate population density by simulation and inverse prediction (Efford 2004; Efford, Dawson & Robbins 2004). A restricted range of SECR models may be fitted (detection functions with more than 2 parameters are not supported, nor are covariates).

Usage

```
ip.secr (capthist, predictorfn = pfn, predictortype = 'null',
    detectfn = 0, mask = NULL, start = NULL, boxsize = 0.1, centre = 3,
    min.nsim = 10, max.nsim = 2000, CVmax = 0.002, var.nsim = 1000,
    maxbox = 5, ...)

pfn(capthist, N.estimator)
```

Arguments

capthist	capthist object including capture data and detector (trap) layout
predictorfn	a function with two arguments (the first a capthist object) that returns a vector of predictor values
predictortype	value (usually character) passed as the second argument of predictorfn
detectfn	numeric code for detection function (0 halfnormal, 2 exponential, 3 uniform)
mask	optional habitat mask to limit simulated population
start	vector of np initial parameter values (density, g0 and sigma)

boxsize	scalar or vector of length np for size of design as fraction of central parameter value
centre	number of centre points in simulation design
min.nsim	minimum number of simulations per point
max.nsim	maximum number of simulations per point
CVmax	tolerance for precision of points in predictor space
var.nsim	number of additional simulations to estimate variance-covariance matrix
maxbox	maximum number of attempts to 'frame' solution
...	further arguments passed to sim.popn
N.estimator	character value indicating population estimator to use

Details

'Inverse prediction' uses methods from multivariate calibration (Brown 1982). The goal is to estimate population density (D) and the parameters of a detection function (usually g0 and sigma) by 'matching' statistics from predictorfn(capthist) (the target vector) and statistics from simulations of a 2-D population using the postulated detection model. Statistics (see Notes) are defined by the predictor function, which should return a vector equal in length to the number of parameters (np = 3). Simulations of the 2-D population use sim.popn. The simulated population is sampled with sim.capthist according to the detector type (e.g., 'single' or 'multi') and detector layout specified in traps(capthist).

... may be used to control aspects of the simulation by passing named arguments (other than D) to sim.popn. The most important arguments of sim.popn to keep an eye on are 'buffer' and 'Ndist'. 'buffer' defines the region over which animals are simulated (unless mask is specified) - the region should be large enough to encompass all animals that might be caught. 'Ndist' controls the number of individuals simulated within the buffered or masked area. The default is 'poisson'. Use 'Ndist = fixed' to fix the number in the buffered or masked area A at $N = DA$. This conditioning reduces the estimated standard error of $\hat{D}$, but conditioning is not always justified - seek advice from a statistician if you are unsure.

The simulated 2-D distribution of animals is Poisson by default. There is no 'even' option as in Density.

Simulations are conducted on a factorial experimental design in parameter space - i.e. at the vertices of a cuboid 'box' centred on the working values of the parameters, plus an optional number of centre points. The size of the 'box' is specified as a fraction of the working values, so for example the limits on the density axis are D*(1–boxsize) and D*(1+boxsize) where D* is the working value of D. For g0, this computation uses the odds transformation (g0/(1–g0)). boxsize may be a vector defining different scaling on each parameter dimension.

A multivariate linear model is fitted to predict each set of simulated statistics from the known parameter values. The number of simulations at each design point is increased (doubled) until the residual standard error divided by the central value is less than CVmax for all parameters. An error occurs if max.nsim is exceeded.

Once a model with sufficient precision has been obtained, a new working vector of parameter estimates is 'predicted' by inverting the linear model and applying it to the target vector. A working vector is accepted as the final estimate when it lies within the box; this reduces the bias from using a linear approximation to extrapolate a nonlinear function. If the working vector lies outside the box then a new design is centred on value for each parameter in the working vector.

Once a final estimate is accepted, further simulations are conducted to estimate the variance-covariance matrix. These also provide a parametric bootstrap sample to evaluate possible bias. Set var.nsim = 0 to suppress the variance step.

See Efford et al. (2004) for another description of the method, and Efford et al. (2005) for an application.

The value of `predictortype` is passed as the second argument of the chosen `predictorfn`. By default this is `pfn`, for which the second argument (`N.estimator`) is a character value from c('n', 'null','zippin','jackknife'), corresponding respectively to the number of individuals caught (Mt+1), and $\hat{N}$ from models M0, Mh and Mb of Otis et al. (1978).

If not provided, the starting values are determined automatically with `autoini`.

Linear measurements are assumed to be in metres and density in animals per hectare (10 000 m^2).

Value

For `ip.secr`, a list comprising

`call`	the function call
`IP`	dataframe with estimated density ha^{-1}, g0 and sigma (m)
`vcov`	variance-covariance matrix of estimates
`ip.nsim`	total number of simulations
`variance.bootstrap`	dataframe summarising simulations for variance estimation
`proctime`	processor time (seconds)

For `pfn`, a vector of numeric values corresponding to $\hat{N}$, $\hat{p}$, and `RPSV`, a measure of the spatial scale of individual detections.

Note

Each statistic is expected to have a monotonic relationship with one parameter when the other parameters are held constant. Typical statistics are -

Statistic	Parameter
$\hat{N}$	D
$\hat{p}$	g_0
$RPSV$	σ

where $\hat{N}$ and $\hat{p}$ are estimates of population size and capture probability from the naive application of a nonspatial population estimator, and $RPSV$ is a trap-revealed measure of the scale of movement.

This method provides nearly unbiased estimates of the detection parameter g0 when data are from single-catch traps (likelihood-based estimates of g0 are biased in this case - Efford, Borchers & Byrom 2009).

The implementation largely follows that in Density, and it may help to consult the Density online help. There are some differences: the M0 and Mb estimates of population-size in `ip.secr` can take non-integer values; the simulation design used by `ip.secr` uses odds(g0) rather than g0; the default boxsize and CVmax differ from those in Density 4.4. There is no provision in `ip.secr` for two-phase estimation, using a different experimental design at the second phase. If you wish you can achieve the same effect by using the estimates as starting values for a second call of `ip.secr` (see examples).

Maximum likelihood estimates from `secr.fit` are preferable in several respects to estimates from inverse prediction (speed*; more complex models; tools for model selection). `ip.secr` is provided for checking estimates of g0 from single-catch traps, and for historical continuity.

* autoini with thin = 1 provides fast estimates from a simple halfnormal model if variances are not required.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Brown, P. J. (1982) Multivariate calibration. *Journal of the Royal Statistical Society, Series B* **44**, 287–321.

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

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

Efford, M. G., Dawson, D. K. and Robbins C. S. (2004) DENSITY: software for analysing capture-recapture data from passive detector arrays. *Animal Biodiversity and Conservation* **27**, 217–228.

Efford, M. G., Warburton, B., Coleman, M. C. and Barker, R. J. (2005) A field test of two methods for density estimation. *Wildlife Society Bulletin* **33**, 731–738.

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**.

See Also

capthist, secr.fit, RPSV, autoini, sim.popn, detection functions

Examples

```
## Not run:
## these calculations may take several minutes

data(captdata)

## default settings
ip.secr (captdata)

## coarse initial fit, no variance step
ip1 <- ip.secr (captdata, boxsize = 0.2, CVmax=0.01, var=0)
## refined fit
ip2 <- ip.secr (captdata, start = ip1$IP[,'estimate'],
    boxsize = 0.1, CVmax=0.002, var=1000)
ip2

## improvise another predictor function (dbar instead of RPSV)
pfn2 <- function (capthist, v) {  ## v is not used
    sumni <- sum(capthist!=0)   ## total detections
    n <- nrow(capthist)         ## number of individuals
    nocc <- ncol(capthist)      ## number of occasions
    c(N = n, p = sumni/n/nocc, dbar = dbar(capthist))
}
ip.secr (captdata, predictorfn = pfn2)

## End(Not run)
```

`LLsurface.secr` *Plot likelihood surface*

Description

Calculate log likelihood over a grid of values of two beta parameters from a fitted secr model and optionally make an approximate contour plot of the log likelihood surface.

Usage

```
LLsurface.secr(object, betapar = c("g0", "sigma"), xval = NULL, yval = NULL,
centre = NULL, realscale = TRUE, plot = TRUE, plotfitted = TRUE, ...)
```

Arguments

`object`	`secr` object output from `secr.fit`
`betapar`	character vector giving the names of two beta parameters
`xval`	vector of numeric values for x-dimension of grid
`yval`	vector of numeric values for y-dimension of grid
`centre`	vector of central values for all beta parameters
`realscale`	logical. If TRUE input and output of x and y is on the untransformed (inverse-link) scale.
`plot`	logical. If TRUE a contour plot is produced
`plotfitted`	logical. If TRUE the MLE from `object` is shown on the plot (+)
`...`	other arguments passed to `contour`

Details

`centre` is set by default to the fitted values of the beta parameters in `object`. This has the effect of holding parameters other than those in `betapar` at their fitted values.

If `xval` or `yval` is not provided then 11 values are set at equal spacing between 0.8 and 1.2 times the values in `centre` (on the 'real' scale if `realscale` = TRUE and on the 'beta' scale otherwise).

Contour plots may be customized by passing graphical parameters through the ... argument.

Value

Invisibly returns a matrix with the log likelihood evaluated at each grid point

Note

`LLsurface.secr` works for named 'beta' parameters rather than 'real' parameters. The default `realscale = TRUE` only works for beta parameters that share the name of the real parameter to which they relate i.e. the beta parameter for the base level of the real parameter. This is because link functions are defined for real parameters not beta parameters.

The contours are approximate because they rely on interpolation. See Examples for a more reliable way to compare the likelihood at the MLE with nearby points on the surface.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
## Not run:
data(secrdemo)
LLsurface.secr(secrdemo.CL, xval = seq(0.16,0.40,0.02),
    yval = 25:35, nlevels = 20)

## now verify MLE
## click on MLE and apparent 'peak'
xy <- locator(2)
temp <- LLsurface.secr(secrdemo.CL, xval = xy$x,
     yval = xy$y, plot = FALSE)
temp

## End(Not run)
```

---

logit *Logit Transformation*

Description

Transform real values to the logit scale.

Usage

```
logit(x)
invlogit(y)
```

Arguments

x	vector of numeric values in (0,1) (possibly a probability)
y	vector of numeric values

Details

The logit transformation is defined as $\text{logit}(x) = \log(\frac{x}{1-x})$ for $x \in (0, 1)$.

Value

Numeric value on requested scale.

Note

`logit` is equivalent to `qlogis`, and `invlogit` is equivalent to `plogis` (both R functions in the **stats** package). `logit` and `invlogit` are used in **secr** because they are slightly more robust to bad input, and their names are more memorable!

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
logit(0.5)
invlogit(logit(0.2))
```

LR.test *Likelihood Ratio Test for SECR Models*

Description

Compute likelihood ratio test to compare two fitted models, one nested within the other.

Usage

```
LR.test(secr1, secr2)
```

Arguments

secr1	fitted secr model
secr2	fitted secr model

Details

The test statistic is twice the difference of the maximized likelihoods. It is compared to a chi-square distribution with df equal to the number of extra parameters in the more complex model.

The models must be nested (no check is performed - this is up to the user), but either secr1 or secr2 may be the more general model.

Value

Object of class 'htest', a list with components

statistic	value the test statistic
parameter	degrees of freedom of the approximate chi-squared distribution of the test statistic
p.value	probability of test statistic assuming chi-square distribution
method	character string indicating the type of test performed
data.name	character string with names of secr models compared

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

AIC.secr, score.test

Examples

```
data(secrdemo)
AIC (secrdemo.0, secrdemo.b)
LR.test  (secrdemo.0, secrdemo.b)
```

`make.capthist` *Construct capthist Object*

Description

Form a `capthist` object from a data frame of capture records and a `traps` object.

Usage

```
make.capthist(captures, traps, fmt = "trapID", noccasions = NULL,
    covnames = NULL, bysession = TRUE, sortrows = TRUE, cutval = NULL,
    tol = 0.01)
```

Arguments

`captures`	dataframe of capture records in one of two possible formats (see Details)
`traps`	object of class `traps` describing an array of passive detectors
`fmt`	character string for capture format. Valid values are 'XY' and 'trapID'.
`noccasions`	number of occasions on which detectors were operated
`covnames`	character vector of names to use for covariate fields if present
`bysession`	logical, if true then ID are made unique by session
`sortrows`	logical, if true then rows are sorted in ascending order of animalID
`cutval`	numeric, threshold of signal strength for 'signal' detector type
`tol`	numeric, tolerance (m) when assigning coordinates for 'transect' detector type

Details

`make.capthist` is the recommended way to prepare data for `secr.fit`. Each row of the input data frame represents a detection on one occasion. The capture data frame may be formed from a text file with `read.captures`.

Input formats are based on the Density software (Efford 2007). If `fmt` = 'XY' the required fields are (session, ID, occasion, x, y) in that order. If `fmt` = 'trapID' the required fields are (session, ID, occasion, trap), where `trap` is the numeric index of the relevant detector in `traps`. `session` and `ID` may be character-, vector- or factor-valued; other required fields are numeric. Fields are matched by position (column number), *not* by name. Columns after the required fields are interpreted as individual covariates that may be continuous (e.g., size) or categorical (e.g., age, sex).

If `captures` has data from multiple sessions then `traps` may be either a list of `traps` objects, one per session, or a single `traps` object that is assumed to apply throughout. Similarly, `noccasions` may be a vector specifying the number of occasions in each session.

Covariates are assumed constant for each individual; the first non-missing value is used. The length of `covnames` should equal the number of covariate fields in `captures`.

`bysession` takes effect when the same individual is detected in two or more sessions: TRUE results in one capture history per session, FALSE has the effect of generating a single capture history (this is not appropriate for the models currently provided in **secr**).

Deaths are coded as negative values in the occasion field of `captures`. Occasions should be numbered 1, 2, ..., noccasions. By default, the number of occasions is the maximum value of 'occasion' in `captures`.

Value

An object of class capthist (a matrix or array of detection data with attributes for detector positions etc.). For 'single' and 'multi' detectors this is a matrix with one row per animal and one column per occasion (dim(capthist)=c(nc,noccasions)); each element is either zero (no detection) or a detector number (the row number in traps *not* the row name). For 'proximity' detectors capthist is an array of values {-1, 0, 1} and dim(capthist)=c(nc,noccasions,ntraps). The number of animals nc is determined from the input, as is noccasions if it is not specified. traps, covariates and other data are retained as attributes of capthist.

Deaths during the experiment are represented as negative values in capthist.

If the input has data from multiple sessions then the output is an object of class c('list','capthist') comprising a list of single-session capthist objects.

Note

make.capthist requires that the data for captures and traps already exist as R objects. To read data from external (text) files, first use read.captures and read.traps.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G. (2007) *Density 4.1: software for spatially explicit capture–recapture*. Department of Zoology, University of Otago, Dunedin, New Zealand. http://www.otago.ac.nz/density.

See Also

capthist, traps, read.captures, secr.fit, sim.capthist

Examples

```
## load demonstration data and peek at XY data
data(rawdata)
captXY[1:5,]
trapXY[1:5,]

demotraps <- read.traps(data = trapXY)
demoCHxy  <- make.capthist (captXY, demotraps, fmt = 'XY')

demoCHxy             ## print method for capthist
plot(demoCHxy)       ## plot method for capthist
summary(demoCHxy)    ## summary method for capthist
```

`make.mask` *Build Habitat Mask*

Description

Construct a habitat mask object for spatially explicit capture-recapture. A mask object is a set of points with optional attributes.

Usage

```
make.mask(traps, buffer = 100, spacing = NULL, nx = 64,
    type = "traprect", poly = NULL, pdotmin = 0.001, ...)
```

Arguments

`traps`	object of class `traps`
`buffer`	width of buffer in metres
`spacing`	spacing between grid points (metres)
`nx`	number of grid points in 'x' direction
`type`	character string for method to use ('traprect', 'trapbuffer', 'pdot', 'polygon')
`poly`	matrix of two columns interpreted as the x and y coordinates of a bounding polygon (optional)
`pdotmin`	minimum detection probability for inclusion in mask when type = 'pdot' (optional)
`...`	additional arguments passed to `pdot` when `type` = 'pdot'

Details

The 'traprect' method constructs a grid of points in the rectangle formed by adding a buffer strip to the minimum and maximum x-y coordinates of the detectors in `traps`. Both 'trapbuffer' and 'pdot' start with a 'traprect' mask and drop some points.

The 'trapbuffer' method restricts the grid to points within distance `buffer` of any detector.

The 'pdot' method restricts the grid to points for which the net detection probability $p.(\mathbf{X})$ (see `pdot`) is at least `pdotmin`. Additional parameters are used by `pdot` (detectpar, noccasions). Set these with the ... argument; otherwise `make.mask` will silently use the arbitrary defaults.

The 'polygon' method places points on a rectangular grid clipped to the polygon (`buffer` is not used).

If `spacing` is not specified then it is determined by dividing the range of the x coordinates (including any buffer) by `nx`.

Value

an object of class `mask`

Note

A warning is displayed if type = 'pdot' and the buffer is too small to include all points with $p.$ > pdotmin.

A habitat mask is needed to fit an SECR model and for some related computations. The default mask settings in `secr.fit` may be good enough, but it is preferable to use `make.mask` to construct a mask in advance and to pass that mask as an argument to `secr.fit`.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`mask`, `subset.mask`, `pdot`

Examples

```
temptrap <- make.grid(nx = 10, ny = 10, spacing = 30)

## default method: traprect
tempmask <- make.mask(temptrap, spacing = 5)
plot(tempmask)
summary (tempmask)

## make irregular detector array by subsampling
## form mask by 'trapbuffer' method
temptrap <- subset (temptrap, sample(nrow(temptrap), size = 30))
tempmask <- make.mask (temptrap, spacing = 5, type = 'trapbuffer')
plot (tempmask)
plot (temptrap, add = TRUE)

## form mask by 'pdot' method
temptrap <- make.grid(nx = 6, ny = 6)
tempmask <- make.mask (temptrap, buffer = 150, type = 'pdot',
    pdotmin = 0.0001, detectpar = list(g0 = 0.1, sigma = 30),
    noccasions = 4)
plot (tempmask)
plot (temptrap, add = TRUE)
```

`make.traps` *Build Detector Array*

Description

Construct a rectangular array of detectors (trapping grid) or a circle of detectors or a polygonal search area.

Usage

```
make.grid(nx = 6, ny = 6, spacex = 20, spacey = 20, spacing = NULL,
    detector = "multi", binomN = 0, originxy = c(0,0), hollow = F,
    ID = 'alphay')

make.circle (n = 20, radius = 100, spacing = NULL,
    detector = "multi", originxy = c(0,0), IDclockwise = T)

make.poly (polylist = NULL, x = c(-50,-50,50,50), y = c(-50,50,50,-50))

make.transect (transectlist = NULL, x = c(-50,-50,50,50),
    y = c(-50,50,50,-50))
```

Arguments

nx	number of columns of detectors
ny	number of rows of detectors
spacex	distance between detectors in 'x' direction (nominally in metres)
spacey	distance between detectors in 'y' direction (nominally in metres)
spacing	distance between detectors (x and y directions)
detector	character value for detector type - 'single', 'multi' etc.
binomN	maximum value when detector == 'count'
originxy	vector origin for x-y coordinates
hollow	logical for hollow grid
ID	character string to control row names
n	number of detectors
radius	radius of circle (nominally in metres)
IDclockwise	logical for numbering of detectors
polylist	list of dataframes with coordinates for polygons
transectlist	list of dataframes with coordinates for transects
x	x coordinates of vertices
y	y coordinates of vertices

Details

make.grid generates coordinates for nx.ny traps at separations spacex and spacey. If spacing is specified it replaces both spacex and spacey. The bottom-left (southwest) corner is at originxy. For a hollow grid, only detectors on the perimeter are retained. By default, identifiers are constructed from a letter code for grid rows and an integer value for grid columns ('A1', 'A2',...). 'Hollow' grids are always numbered clockwise in sequence from the bottom-left corner. Other values of ID have the following effects:

ID	Effect
numx	column-dominant numeric sequence
numy	row-dominant numeric sequence
numxb	column-dominant boustrophedonical numeric sequence (try it!)
numyb	row-dominant boustrophedonical numeric sequence

alphax	column-dominant alphanumeric
alphay	row-dominant alphanumeric

`make.circle` generates coordinates for n traps in a circle centred on `originxy`. If `spacing` is specified then it overrides the `radius` setting; the radius is adjusted to provide the requested straightline distance between adjacent detectors. Traps are numbered from the trap due east of the origin, either clockwise or anticlockwise as set by `IDclockwise`.

Polygon vertices may be specified with `x` and `y` in the case of a single polygon, or as `polylist` for one or more polygons. Each component of `polylist` is a dataframe with columns 'x' and 'y'. `polylist` takes precedence. `make.poly` automatically closes the polygon by repeating the first vertex if the first and last vertices differ.

Transects are defined by a sequence of vertices as for polygons, except that they are not closed.

Value

An object of class `traps` comprising a data frame of x- and y-coordinates, the detector type ('single', 'multi', or 'proximity' etc.), and possibly other attributes.

Note

Several methods are provided for manipulating detector arrays - see `traps`.

Author(s)

Murray Efford `<murray.efford@otago.ac.nz>`

References

Efford, M.G. (2007) *Density 4.1: software for spatially explicit capture–recapture*. Department of Zoology, 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 and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer, New York. Pp. 255–269.

See Also

`read.traps`,`detector`, `print.traps`, `plot.traps`, `traps`

Examples

```
demo.traps <- make.grid()
plot(demo.traps)

## compare numbering schemes
par (mfrow = c(2,4), mar = c(1,1,1,1), xpd = TRUE)
for (id in c('numx', 'numy', 'alphax', 'alphay', 'numxb',
    'numyb'))
{
    temptrap <- make.grid(nx = 7, ny = 5, ID = id)
    plot (temptrap, border = 10, lab = TRUE, offset = 7,
        gridl = FALSE)
}
```

```
temptrap <- make.grid(nx = 7, ny = 5, hollow = TRUE)
plot (temptrap, border = 10, lab = TRUE, gridl = FALSE)

plot(make.circle(n = 20, spacing = 30), lab = TRUE, offset = 9)
summary(make.circle(n = 20, spacing = 30))
```

`mask` *Mask Object*

Description

Encapsulate a habitat mask for spatially explicit capture–recapture.

Details

A habitat mask serves four main purposes in spatially explicit capture–recapture. Firstly, it defines an outer limit to the area of integration; habitat beyond the mask may be occupied, but animals there should have negligible chance of being detected (see `pdot` and below). Secondly, it distinguishes sites in the vicinity of the detector array that are 'habitat' (i.e. have the potential to be occupied) from 'non-habitat'. Thirdly, it discretizes continuous habitat as a list of points. Each point is notionally associated with a cell (pixel) of uniform density. Discretization allows the SECR likelihood to be evaluated by summing over grid cells. Fourthly, the x-y coordinates of the mask and any habitat covariates may be used to build spatial models of density. For example, a continuous or categorical habitat covariate 'cover' measured at each point on the mask might be used in a formula for density such as D = ~cover.

In relation to the first purpose, the definition of 'negligible' is fluid. Any probability less than 0.01 seems OK in the sense of not causing noticeable bias in density estimates, but extent of the mask affects the binomial sampling variance of density derived from conditional likelihood estimates of the detection function (M. Efford unpubl. results).

Mask points are stored in a data frame with columns 'x' and 'y'. The number of rows equals the number of points.

Possible mask attributes –

type	'traprect', 'trapbuffer', 'pdot', 'polygon' (see make.mask) or 'user'
polygon	vertices of polygon defining habitat boundary, for type = 'polygon'
pdotmin	threshold of $p.(\mathbf{X})$ for type = 'pdot'
covariates	dataframe of site-specific covariates
meanSD	data frame with centroid (mean and SD) of x and y coordinates
area	area (ha) of the grid cell associated with each point
spacing	nominal spacing (metres) between adjacent points
boundingbox	data frame of 4 rows, the vertices of the bounding box of all grid cells in the mask

Attributes other than `covariates` are generated automatically by `make.mask`. Type 'user' refers to masks input from a text file with `read.mask`.

Note

A habitat mask is needed by `secr.fit`, but one will be generated automatically if none is provided. You should be aware of this and check that the default settings (e.g. `buffer`) are appropriate.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`make.mask`, `read.mask`, `secr.fit`, `secr density models`

`model.average` *Model averaging for SECR Models*

Description

AICc-weighted average of estimated 'real' or 'beta' parameters from multiple fitted secr models.

Usage

```
model.average(..., realnames = NULL, betanames = NULL, newdata = NULL,
    alpha = 0.05, dmax = 10, covar = FALSE, average = 'link')

collate (..., realnames = NULL, betanames = NULL, newdata = NULL,
    scaled = FALSE, alpha = 0.05, perm = 1:4, fields = 1:4)
```

Arguments

`...`	`secr` objects
`realnames`	character vector of real parameter names
`betanames`	character vector of beta parameter names
`newdata`	optional dataframe of values at which to evaluate models
`scaled`	logical for scaling of sigma and g0 (see Details)
`alpha`	alpha level for confidence intervals
`dmax`	numeric, the maximum AIC difference for inclusion in confidence set
`covar`	logical, if TRUE then return variance-covariance matrix
`average`	character string for scale on which to average real parameters
`perm`	permutation of dimensions in output from `collate`
`fields`	vector to restrict summary fields in output

Details

Models to be compared must have been fitted to the same data and use the same likelihood method (full vs conditional). If `realnames` == NULL and `betanames` == NULL then all real parameters will be averaged; in this case all models must use the same real parameters. To average beta parameters, specify `betanames` (this is ignored if a value is provided for `realnames`). See `predict.secr` for an explanation of the optional argument `newdata`; `newdata` is ignored when averaging beta parameters.

Model-averaged estimates for parameter θ are given by

$$\hat{\theta} = \sum_k w_k \hat{\theta}_k$$

where the subscript k refers to a specific model and the w_k are AIC weights with small sample adjustment (see `AIC.secr` for details). Averaging of real parameters may be done on the link scale before back-transformation (`average='link'`) or after back-transformation (`average='real'`).

Models for which dAICc > `dmax` are given a weight of zero and effectively are excluded from averaging.

Also,

$$\text{var}(\hat{\theta}) = \sum_k w_k (\text{var}(\hat{\theta}_k | \beta_k) + \beta_k^2)$$

where $\hat{\beta}_k = \hat{\theta}_k - \hat{\theta}$ and the variances are asymptotic estimates from fitting each model k. This follows Burnham and Anderson (2004) rather than Buckland et al. (1997).

`collate` extracts parameter estimates from a set of fitted secr model objects. `fields` may be used to select a subset of summary fields ('estimate','SE.estimate','lcl','ucl') by name or number.

The argument `scaled` applies only to the detection parameters g0 and sigma, and only to models fitted with `scalesigma` or `scaleg0` switched on. If `scaled` is TRUE then each estimate is multiplied by its scale factor (1/D^0.5 and 1/sigma^2 respectively).

Value

For `model.average`, an array of model-averaged estimates, their standard errors, and a $100(1-\alpha)\%$ confidence interval. The interval for real parameters is backtransformed from the link scale. If there is only one row in `newdata` or beta parameters are averaged or averaging is requested for only one parameter then the array is collapsed to a matrix. If `covar = TRUE` then a list is returned with separate components for the estimates and the variance-covariance matrices.

For `collate`, a 4-dimensional array of model-specific parameter estimates. By default, the dimensions correspond respectively to rows in `newdata` (usually sessions), models, statistic fields (estimate, SE.estimate, lcl, ucl), and parameters ('D', 'g0' etc.). For particular comparisons it often helps to reorder the dimensions with the `perm` argument.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Buckland S. T., Burnham K. P. and Augustin, N. H. (1997) Model selection: an integral part of inference. *Biometrics* **53**, 603–618.

Burnham, K. P. and Anderson, D. R. (2002) *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Second edition. New York: Springer-Verlag.

Burnham, K. P. and Anderson, D. R. (2004) Multimodel inference - understanding AIC and BIC in model selection. *Sociological Methods & Research* **33**, 261–304.

See Also

AIC.secr, secr.fit

Examples

```
## Compare two models fitted previously
## secrdemo.0 is a null model
## secrdemo.b has a learned trap response

data(secrdemo)
model.average(secrdemo.0, secrdemo.b)
model.average(secrdemo.0, secrdemo.b, betanames = c('D','g0','sigma'))

## In this case we find the difference was actually trivial...
## (subscripting of output is equivalent to setting fields = 1)

collate (secrdemo.0, secrdemo.b, perm = c(4,2,3,1))[,,1,]
```

---

ovenbird *Ovenbird Mist-netting Dataset*

Description

Data from a multi-year mist-netting study of ovenbirds (*Seiurus aurocapilla*) at a site in Maryland, USA.

Usage

```
data(ovenbird)
```

Details

From 2005 to 2009 D. K. Dawson and M. G. Efford conducted a capture–recapture survey of breeding birds in deciduous forest at the Patuxent Research Refuge near Laurel, Maryland, USA. The forest was described by Stamm, Davis & Robbins (1960), and has changed little since. Analyses of data from previous mist-netting at the site by Chan Robbins were described in Efford, Dawson & Robbins (2004) and Borchers & Efford (2008).

Forty-four mist nets (12 m long, 30-mm mesh) spaced 30 m apart on the perimeter of a 600-m x 100-m rectangle were operated for approximately 9 hours on each of 9 or 10 non-consecutive days during late May and June in each year. Netting was passive (i.e. song playback was not used to lure birds into the nets). Birds received individually numbered bands, and both newly banded and previously banded birds were released at the net where captured. Sex was determined in the hand from the presence of a brood patch (females) or cloacal protuberance (males). A small amount of extra netting was done by other researchers after the main session in some years.

This dataset comprises all records of adult (after-hatch-year) ovenbirds caught during the main session in each of the five years 2005–2009. One ovenbird was killed by a predator in the net in 2009, as indicated by a negative net number in the dataset. Sex was determined in the hand from

the presence of a brood patch (females) or cloacal protuberance (males). Birds are listed by their band number (4-digit prefix, '.', and 5-digit number). Recaptures within a day are not included in this dataset, so each bird occurs at most once per day and the detector type is 'multi' rather than 'proximity'. Although several individuals were captured in more than one year, no use is made of this information in the analyses presently offered in **secr**.

The data are provided as a multi-session `capthist` object 'ovenCH'. Sex is coded as a categorical individual covariate ('M' or 'F').

An analysis of the data for males in the first four years showed that they tended to avoid nets after their first capture within a season (Dawson & Efford in press). While the species was present consistently, the number of detections in any one year was too small to give reliable estimates of density; pooling of detection parameters across years helped to improve precision.

Source

D. K. Dawson (<ddawson@usgs.gov>) and M. G. Efford unpublished data.

References

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

Dawson, D. K. and Efford, M. G. (2009) Bird population density estimated from acoustic signals. *Journal of Applied Ecology* **46**, 1201–1209.

Efford, M. G., Dawson, D. K. and Robbins C. S. (2004) DENSITY: software for analysing capture-recapture data from passive detector arrays. *Animal Biodiversity and Conservation* **27**, 217–228.

Stamm, D. D., Davis, D. E. and Robbins, C. S. (1960) A method of studying wild bird populations by mist-netting and banding. *Bird-Banding* **31**, 115–130.

See Also

capthist

Examples

```
## Not run:

## commands used to create ovenCH from the input files
## 'netsites0509.txt' and 'ovencapt.txt'
## for information only - these files not distributed
netsites0509 <- read.traps(file = 'netsites0509.txt',
    skip = 1, detector = 'multi')
temp <- read.captures('ovencapt.txt', colClasses=c('character',
    'character', 'numeric', 'numeric', 'character'))
ovenCH <- make.capthist(temp, netsites0509, covnames=c('Sex','Age'))


## End(Not run)

require (graphics)
data(ovenbird)
par(mfrow = c(1,5), mar = c(1,1,4,1))
plot(ovenCH, tracks = TRUE, varycol = TRUE)
```

```
counts(ovenCH, 'n')

## Not run:

## array constant over years, so build mask only once
ovenmask <- make.mask(traps(ovenCH)[['2005']], type='pdot', buffer=400,
    spacing=15, detectpar=list(g0=0.03, sigma=90), nocc=10)

## fit constant-density model
ovenbird.model.1 <- secr.fit(ovenCH, mask = ovenmask)
ovenbird.model.1

## fit net avoidance model
ovenbird.model.1b <- secr.fit(ovenCH, mask = ovenmask, model =
    list(g0~b))
ovenbird.model.1b


## End(Not run)

## compare & average pre-fitted models
AIC(ovenbird.model.1, ovenbird.model.1b)
model.average(ovenbird.model.1, ovenbird.model.1b)

## select one year to plot
plot(ovenbird.model.1b, newdata = data.frame(session = '2005',
    b = 0))
```

`ovensong` *Ovenbird Acoustic Dataset*

Description

Data from an acoustic survey of ovenbirds (*Seiurus aurocapilla*) at a site in Maryland, USA.

Usage

```
data(ovensong)
```

Details

In June 2007 D. K. Dawson and M. G. Efford used a moving 4-microphone array to survey breeding birds in deciduous forest at the Patuxent Research Refuge near Laurel, Maryland, USA. The data for ovenbirds were used to demonstrate a new method for analysing acoustic data (Dawson and Efford 2009). See `ovenbird` for mist-netting data from the same site over 2005–2009, and for other background.

Over five days, four microphones were placed in a square (21-m side) centred at each of 75 points in a rectangular grid (spacing 50 m); on each day points 100 m apart were sampled sequentially. Recordings of 5 minutes duration were made in .wav format on a 4-channel digital sound recorder.

The data are estimates of average power on each channel (microphone) for the first song of each ovenbird distinguishable in a particular 5-minute recording. Power was estimated with the sound

analysis software Raven Pro 1.4 (Charif et al. 2008), using a window of 0.7 s duration and frequencies between 4200 and 5200 Hz, placed manually at the approximate centre of each ovenbird song. Sometimes this frequency range was obscured by insect noise so an alternative 1000-Hz range was measured and the values were adjusted by regression.

The data are provided as a single-session, single-occasion `capthist` object `signalCH`. The 'signal' attribute contains the power measurement in decibels for each detected sound on each channel where the power threshold is exceeded. As the threshold signal (attribute `cutval` = 35) is less than any signal value in this dataset, all detection histories are complete (1,1,1,1) across microphones. For analysis Dawson and Efford applied a higher threshold that treated weaker signals as 'not detected' (see Examples).

The row names of `signalCH` (e.g. '3755AX') are formed from a 4-digit number indicating the sampling location (one of 75 points on a 50-m grid) and a letter A–D to distinguish individual ovenbirds within a 5-minute recording; 'X' indicates power values adjusted by regression.

The default model for sound attenuation is a log-linear decline with distance from the source (linear decline on dB scale). Including a spherical spreading term in the sound attenuation model causes the likelihood surface to become multimodal in this case. Newton-Raphson, the default maximization method in `secr.fit`, is particularly inclined to settle on a local maximum; in the example below we use a set of starting values that have been found by trial and error to yield the global maximum.

Source

D. K. Dawson (<ddawson@usgs.gov>) and M. G. Efford unpublished data.

References

Charif, R. A., Waack, A. M. and Strickman, L. M. (2008) Raven Pro 1.3 User's Manual. Cornell Laboratory of Ornithology, Ithaca, New York.

Dawson, D. K. and Efford, M. G. (2009) Bird population density estimated from acoustic signals. *Journal of Applied Ecology* **46**, 1201–1209.

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.

See Also

`capthist`, `ovenbird`, `detection functions`

Examples

```
data(ovensong)
summary(signalCH)
traps(signalCH)
signal(signalCH)

## apply signal threshold
signalCH.525 <- subset(signalCH, cutval = 52.5)

## Not run:
## models with and without spherical spreading
omask <- make.mask(traps(signalCH), buffer = 200)
ostart <- c(log(20), 80, log(0.1), log(2))
ovensong.model.1 <- secr.fit( signalCH.525, mask = omask,
    start = ostart, detectfn = 11 )
```

```
ovensong.model.2 <- secr.fit( signalCH.525, mask = omask,
    start = ostart, detectfn = 10 )

## End(Not run)

## compare fit of models
AIC(ovensong.model.1, ovensong.model.2)

## density estimates, dividing by 75 to allow for replication
collate(ovensong.model.1, ovensong.model.2)[1,,,'D']/75

## plot attenuation curves cf Dawson & Efford (2009) Fig 5
pars1 <- predict(ovensong.model.1)[c('beta0', 'beta1'), 'estimate']
pars2 <- predict(ovensong.model.2)[c('beta0', 'beta1'), 'estimate']
attenuationplot(pars1, xval=0:150, spherical = TRUE, ylim = c(40,110))
attenuationplot(pars2, xval=0:150, spherical = FALSE, add = TRUE,
    col = 'red')
## spherical spreading only
pars1[2] <- 0
attenuationplot(pars1, xval=0:150, spherical = TRUE, add = TRUE, lty=2)
```

`pdot` *Net Detection Probability*

Description

Compute spatially explicit net probability of detection for individual(s) at given coordinates.

Usage

```
pdot(X, traps, detectfn = 0, detectpar = list(g0 = 0.2, sigma = 20),
    noccasions = 5)
```

Arguments

`X`	coordinates
`traps`	`traps` object
`detectfn`	integer code for detection function q.v.
`detectpar`	a list giving a value for each named parameter of detection function
`noccasions`	number of intervals (occasions)

Details

The probability computed is $p.(\mathbf{X}) = 1 - \prod_k \{1 - p_s(\mathbf{X}, k)\}^S$ where the product is over the detectors in `traps`. The per-occasion detection function p_s is by default half-normal, and is assumed not to vary over the S occasions.

The calculation is not valid for single-catch traps because $p.(\mathbf{X})$ is reduced by competition between animals.

Value

A vector of probabilities, one for each row in X.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

secr, make.mask, detection functions

Examples

```
temptrap <- make.grid()
## per-session detection probability for an individual centred
## at a corner trap. By default, noccasions = 5.
pdot (c(0,0), temptrap, detectpar = list(g0 = 0.2, sigma = 20))
```

---

plot.capthist *Plot Detection Histories*

Description

Display a plot of detection (capture) histories over a map of the detectors.

Usage

```
## S3 method for class 'capthist':
plot(x, rad = 5, hidetraps = FALSE, tracks = FALSE,
    title = TRUE, subtitle = TRUE, add = FALSE, varycol = TRUE,
    icolours = NULL, randcol = FALSE,
    lab1cap = FALSE, laboffset = 4, ncap = FALSE,
    splitocc = NULL, col2 = "green",
    cappar = list(cex = 1.3, pch = 16, col = "blue"),
    trkpar = list(col = "blue", lwd = 1),
    labpar = list(cex = 0.7, col = "black"), ...)
```

Arguments

x	an object of class capthist
rad	radial displacement of dot indicating each capture event from the detector location (used to separate overlapping points)
hidetraps	logical indicating whether trap locations should be displayed
tracks	logical indicating whether consecutive locations of individual animals should be joined by a line
title	logical or character string for title
subtitle	logical or character string for subtitle
add	logical for whether to add to existing plot
varycol	logical for whether to distinguish individuals by colour

icolours	vector of individual colours (when varycol = TRUE)
randcol	logical to use random colours (varycol = TRUE)
lab1cap	logical for whether to label the first capture of each animal
laboffset	distance by which to offset labels from points
ncap	logical to display the number of detections per trap per occasion
splitocc	optional occasion from which second colour is to be used
col2	second colour (used with splitocc)
cappar	list of named graphical parameters for detections (passed to par)
trkpar	list of named graphical parameters for tracks (passed to par)
labpar	list of named graphical parameters for labels (passed to par)
...	arguments to be passed to lines if tracks are plotted

Details

A plot is generated in the style of Density (Efford 2007) using eqscplot from the MASS library.

If title = FALSE no title is displayed; if title = TRUE, the session identifer is used for the title.

If subtitle = FALSE no subtitle is displayed; if subtitle = TRUE, the subtitle gives the numbers of occasions, detections and individuals.

If icolours = NULL and varycol = TRUE then a vector of colours is generated automatically as terrain.colors((nrow(x)+1) * 1.5). If there are too few values in icolours for the number of individuals then colours will be re-used.

If x is a multi-session capthist object then a separate plot is produced for each session. Use par(mfrow = c(nr, nc)) to allow a grid of plots to be displayed simultaneously (nr rows x nc columns).

Value

The number of detections in x.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G. (2007) *Density 4.1: software for spatially explicit capture–recapture*. Department of Zoology, University of Otago, Dunedin, New Zealand. http://www.otago.ac.nz/density

See Also

capthist

Examples

```
demotrap <- make.grid()
tempcapt <- sim.capthist(demotrap,
    popn = list(D = 5, buffer = 50),
    detectpar = list(g0 = 0.15, sigma = 30))
plot(tempcapt, border = 10, rad = 3, tracks = TRUE,
    lab1cap = TRUE, laboffset = 2.5)
```

plot.mask *Plot Habitat Mask*

Description

Plot a habitat mask either as points or as an image plot. Colours may be used to show the value of one mask covariate.

Usage

```
## S3 method for class 'mask':
plot(x, border = 20, add = F, covariate = NULL, axes = F,
    dots = T, col = "grey", ppoly = T, polycol = "red", ...)
```

Arguments

x	mask object
border	width of blank display border (metres)
add	logical for adding mask points to an existing plot
covariate	name (as character string in quotes) or column number of a covariate to use for colouring
axes	logical for plotting axes
dots	logical for plotting mask points as dots, rather than as image plot
col	colour(s) to use for plotting
ppoly	logical for whether the bounding polygon should be plotted (applies only if mask type = 'polygon')
polycol	colour for outline of polygon (ppoly = TRUE)
...	other arguments passed to eqscplot

Details

The argument dots selects between two distinct types of plot; see image for more on image plots. If using a covariate to colour points, the col argument should be a colour vector of length > 1; the default is heat.colors(12).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

colours, mask

Examples

```
# simple

temptrap <- make.grid()
tempmask <- make.mask(temptrap)
plot (tempmask)

## restrict to points over an arbitrary detection threshold,
## add covariate, plot image and overlay traps

tempmask <- subset(tempmask, pdot(tempmask,temptrap)>0.001)

covariates (tempmask) <- data.frame(circle =
    exp(-(tempmask$x^2 + tempmask$y^2)/10000) )

plot (tempmask, covariate = 'circle', dots = FALSE, axes = TRUE,
    add = TRUE, col = terrain.colors(256))
plot (temptrap, add = TRUE)

## add a legend

par(cex = 0.9)
covrange <- range(covariates(tempmask)$circle)
step <- diff(covrange)/8
colourlev <- terrain.colors(256)[seq(1,256,(256-1)/8)]
zlev <- formatC(seq(covrange[1],covrange[2],step), format='f',
    digits=2, width=4)
legend (x = 'topright', fill = colourlev, legend = zlev,
    y.intersp = 0.8, title = 'Covariate')

title('Colour mask points with p.(X) > 0.001')
mtext(side=3,line=-1, 'g0 = 0.2, sigma = 20, nocc = 5')
```

---

plot.popn *Plot popn Object*

Description

Display animal locations from a `popn` object.

Usage

```
## S3 method for class 'popn':
plot(x, add = FALSE, frame = TRUE, ...)
```

Arguments

`x`	object of class `popn`
`add`	logical to add points to an existing plot
`frame`	logical to add frame within which points were simulated
`...`	arguments passed to `eqscplot` and `points`

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

popn, sim.popn

Examples

```
temppopn <- sim.popn(D = 5, expand.grid(x = c(0,100), y = c(0,100)))
plot(temppopn, pch = 16, col = 'blue')
```

---

plot.secr *Plot Detection Functions*

Description

Plot detection functions using estimates of parameters in an secr object, or as provided by the user.

Usage

```
## S3 method for class 'secr':
plot(x, newdata = NULL, add = FALSE,
    sigmatick = FALSE, rgr = FALSE, limits = TRUE, alpha = 0.05,
    xval = 0:200, ylim = NULL, xlab = NULL, ylab = NULL, ...)

detectfnplot (detectfn, pars, details = NULL, add = FALSE,
    sigmatick = FALSE, rgr = FALSE, xval = 0:200, ylim = NULL,
    xlab = NULL, ylab = NULL, ...)

attenuationplot (pars, add = FALSE, spherical = TRUE,
    xval = 0:200, ylim = NULL, xlab = NULL, ylab = NULL, ...)
```

Arguments

x	an secr object
newdata	dataframe of data to form estimates
add	logical to add curve(s) to an existing plot
sigmatick	logical; if TRUE the scale parameter sigma is shown by a vertical line
rgr	logical; if TRUE a scaled curve r.g(r) is plotted instead of g(r)
limits	logical; if TRUE pointwise confidence limits are drawn
alpha	alpha level for confidence intervals
xval	vector of distances at for which detection to be plotted
ylim	vector length 2 giving limits of y axis
xlab	label for x axis

ylab	label for y axis
...	arguments to pass to lines
detectfn	integer code for detection function 0 halfnormal, 1 hazard-rate etc.
pars	vector or matrix of parameter values
details	list of ancillary parameters
spherical	logical for whether to include spherical spreading term

Details

newdata is usually NULL, in which case one curve is plotted for each session and group. Otherwise, predict.secr is used to form estimates and plot a curve for each row in newdata.

If axis labels are not provided they default to 'Distance (m)' and 'Detection probability' or 'Detection lambda'.

detectfnplot is an alternative in which the user nominates the type of function and provides parameter values. If pars is a matrix then a separate curve is plotted with the parameter values in each row.

For detectfnplot the signal threshold parameters 'cutval' and 'spherical' should be provided in details (see examples).

Approximate confidence limits for g(r) are calculated using a numerical first-order delta-method approximation to the standard error at each xval. The distribution is assumed to be normal on the logit scale; limits are back-transformed from that scale.

attenuationplot plots the expected decline in signal strength with distance, given parameters β_0 and β_1 for a log-linear model of sound attenuation.

Value

plot.secr invisibly returns a dataframe of the plotted values (or a list of dataframes in the case that newdata has more than one row).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

detection functions, plot, secr

Examples

```
data(secrdemo)
plot (secrdemo.b, xval = 0:100, ylim = c(0, 0.4))
## Add recapture probability
plot (secrdemo.b, newdata = data.frame(b = 1), add = TRUE,
    col='red')

## signal strength detection: 70dB at source, attenuation
## 0.3dB/m, sdS 5dB; detection threshold 40 dB.
detectfnplot (detectfn = 10, c(70, -0.3, 5), details =
    list(cutval = 40))
```

```
## add a function with louder source and spherical spreading...
detectfnplot (detectfn = 11, c(110, -0.3, 5), details =
    list(cutval = 40), add = TRUE, col = 'red')

## matching sound attenuation curves; 'spherical-only' dashed line
attenuationplot (c(70, -0.3), spherical = FALSE, ylim=c(-10,110))
attenuationplot (c(110, 0), spherical = TRUE, add=TRUE, lty=2)
attenuationplot (c(110, -0.3), spherical = TRUE, add = TRUE,
    col = 'red')
```

`plot.traps` *Plot traps Object*

Description

Map the locations of detectors (traps).

Usage

```
## S3 method for class 'traps':
plot(x, border = 100, label = FALSE, offset = c(6,6), add = FALSE,
  hidetr = FALSE, detpar = list(), txtpar = list(), bg = "white",
  gridlines = TRUE, gridspace = 100, gridcol = "grey",
  markvarying = FALSE, ...)
```

Arguments

`x`	a `traps` object
`border`	width of blank margin around the outermost detectors
`label`	logical indicating whether a text label should appear by each detector
`offset`	vector displacement of label from point on x and y axes
`add`	logical to add detectors to an existing plot
`hidetr`	logical to suppress plotting of detectors
`detpar`	list of named graphical parameters for detectors (passed to `par`)
`txtpar`	list of named graphical parameters for labels (passed to `par`)
`bg`	background colour
`gridlines`	logical for plotting grid lines
`gridspace`	spacing of gridlines
`gridcol`	colour of gridlines
`markvarying`	logical to distinguish detectors whose usage varies among ocasions
`...`	arguments to pass to `eqscplot`

Details

`offset` may also be a scalar value for equal displacement on the x and y axes. The `hidetr` option is most likely to be used when plot.traps is called by plot.capthist. See `par` and `colours` for more information on setting graphical parameters. The initial values of graphical parameters are restored on exit.

Axes are not labeled. Use `axis` and `mtext` if necessary.

Value

None

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

plot, traps

Examples

```
temptrap <- make.grid()
plot (temptrap, detpar = list(pch = 16, col = 'blue'),
   label = TRUE, offset = 7)
```

---

popn *Population Object*

Description

Encapsulate the locations of a set of individual animals.

Details

An object of class popn records the locations of a set of individuals, together with ancillary data such as their sex. Often used for a realisation of a spatial point process (e.g. homogeneous Poisson) with known density (intensity). Locations are stored in a data frame with columns 'x' and 'y'.

A popn object has attributes

covariates	data frame with numeric, factor or character variables to be used as individual covariates
model2D	2-D distribution ('poisson', 'cluster', 'IHP')
Ndist	distribution of number of individuals ('poisson', 'fixed')
boundingbox	data frame of 4 rows, the vertices of the rectangular area

The number of rows in covariates must match the length of x and y. See sim.popn for more information on Ndist and model2D.

Note

The popn class is used only occasionally: it is not central to spatially explicit capture recapture.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

sim.popn, plot.popn

possum *Brushtail Possum Trapping Dataset*

Description

Data from a trapping study of brushtail possums at Waitarere, North Island, New Zealand.

Usage

```
data(possum)
```

Details

Brushtail possums (*Trichosurus vulpecula*) are an unwanted invasive species in New Zealand. Although most abundant in forests, where they occasionally exceed densities of 15 / ha, possums live wherever there are palatable food plants and shelter.

Efford et al. (2005) reported a live-trapping study of possums in *Pinus radiata* plantation on coastal sand dunes. The 294-ha site at Waitarere in the North Island of New Zealand was a peninsula, bounded on one side by the sea and on two other sides by the Manawatu river. Cage traps were set in groups of 36 at 20-m spacing around the perimeter of five squares, each 180 m on a side. The squares ('hollow grids') were centred at random points within the 294-ha area. Animals were tagged and released daily for 5 days in April 2002. Subsequently, leg-hold trapping was conducted on a trapping web centred on each square (data not reported here), and strenuous efforts were made to remove all possums by cyanide poisoning and further leghold trapping across the entire area. This yielded a density estimate of 2.26 possums / ha.

Traps could catch at most one animal per day. The live-trapped animals comprised 46 adult females, 33 adult males, 10 immature females and 11 immature males; sex and/or age were not recorded for 4 individuals (M. Coleman unpubl. data). One female possum was twice captured at two sites on one day, having entered a second trap after being released; one record in each pair was selected arbitrarily and discarded.

The data are provided as a single-session `capthist` object 'possumCH'. 'possummask' is a matching mask object - see Examples. Two fitted models ('possum.model.1' & 'possum.model.1b') are provided for illustration.

Source

Landcare Research, New Zealand.

References

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

Efford, M. G., Dawson, D. K. and Robbins C. S. (2004) DENSITY: software for analysing capture-recapture data from passive detector arrays. *Animal Biodiversity and Conservation* **27**, 217–228.

Efford, M. G., Warburton, B., Coleman, M. C. and Barker, R. J. (2005) A field test of two methods for density estimation. *Wildlife Society Bulletin* **33**, 731–738.

See Also

`capthist`

Examples

```
## Not run:

setwd('d:\density communication\webtest\foxton\')
possumtraps <- read.traps(file = 'foxtraps.txt', detector = 'single')
temp <- read.captures('foxton.txt', colClasses=c('character',
    'character', 'numeric', 'character'))
## drop within-day duplicates of animal 5861
temp <- temp[-c(184,186),]

possumCH <- make.capthist(temp, possumtraps)
possummask <- make.mask(possumtraps, buffer = 300, type='pdot',
    pdotmin = 0.001, detectpar = list(g0=0.2, sigma=60), spacing = 10)

## fit constant-density model
possum.model.1 <- secr.fit(possumCH, mask = possummask)
## fit learned trap response model
possum.model.1b <- secr.fit(possumCH, mask = possummask, model = list(g0~b))


## End(Not run)

require (graphics)
data(possum)
plot(possummask)
plot(possumCH, tracks = TRUE, add = TRUE)
plot(traps(possumCH), add = TRUE)
summary(possumCH)

## compare & average pre-fitted models
AIC(possum.model.1, possum.model.1b)
model.average(possum.model.1, possum.model.1b)
```

`predict.secr` *SECR Model Predictions*

Description

Evaluate a spatially explicit capture–recapture model. That is, compute the 'real' parameters corresponding to the 'beta' parameters of a fitted model for arbitrary levels of any variables in the linear predictor.

Usage

```
## S3 method for class 'secr':
predict(object, newdata = NULL, se.fit = TRUE, alpha = 0.05,
    savenew = FALSE, scaled = FALSE, ...)
```

Arguments

`object`	`secr` object output from `secr.fit`

newdata	optional dataframe of values at which to evaluate model
se.fit	logical for whether output should include SE and confidence intervals
alpha	alpha level for confidence intervals
savenew	logical for whether newdata should be saved
scaled	logical for scaling of sigma and g0 (see Details)
...	other arguments

Details

The variables in the various linear predictors are described in secr models and listed for the particular model in the vars component of object.

Optional newdata should be a dataframe with a column for each of the variables in the model (see 'vars' component of object). If newdata is missing then a dataframe is constructed automatically. Default newdata are for a naive animal on the first occasion; numeric covariates are set to zero and factor covariates to their base (first) level.

Standard errors are by the delta method (Lebreton et al. 1992). Confidence intervals are backtransformed from the link scale.

The argument scaled applies only to the detection parameters g0 and sigma, and only to models fitted with scalesigma or scaleg0 switched on. If scaled is TRUE then each estimate is multiplied by its scale factor (1/D^0.5 and 1/sigma^2 respectively).

The value of newdata is optionally saved as an attribute.

Value

When se.fit = FALSE, a dataframe identical to newdata except for the addition of one column for each 'real' parameter. Otherwise, a list with one component for each row in newdata. Each component is a dataframe with one row for each 'real' parameter (density, g0, sigma, b) and columns as below

link	link function
estimate	estimate of real parameter
SE.estimate	standard error of the estimate
lcl	lower 100(1–alpha)% confidence limit
ucl	upper 100(1–alpha)% confidence limit

When newdata has only one row, the structure of the list is 'dissolved' and the return value is one data frame.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

See Also

secr.fit

Examples

```
## load previously fitted secr model with trap response
## and extract estimates of 'real' parameters for both
## naive (b = 0) and previously captured (b = 1) animals

data(secrdemo)
predict (secrdemo.b, newdata = data.frame(b=0:1))

temp <- predict (secrdemo.b, newdata = data.frame(b=0:1),
    save = TRUE)
attr(temp, 'newdata')
```

---

print.capthist *Print Detections*

Description

Print method for capthist objects.

Usage

```
## S3 method for class 'capthist':
print(x, ..., condense = FALSE, sortrows = FALSE)
```

Arguments

x	capthist object
...	arguments to pass to print.default
condense	logical, if true then use condensed format for 3-D data
sortrows	logical, if true then sort output by animal

Details

The condense option may be used to format data from proximity detectors in a slightly more readable form. Each row then presents the detections of an individual in a particular trap, dropping rows (traps) at which the particular animal was not detected.

Value

Invisibly returns a dataframe (condense = TRUE) or array in the format printed.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

print, capthist

Examples

```
## simulated detections of simulated default population of 5/ha
print(sim.capthist(make.grid(nx=5,ny=3)))
```

---

print.mask *Print Habitat Mask*

Description

Print the x-y coordinates of points in a mask object

Usage

```
## S3 method for class 'mask':
print(x, ...)
```

Arguments

x	mask object
...	arguments passed to other functions

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

mask, summary.mask

Examples

```
tempmask <- make.mask(make.grid())
print(tempmask)
```

`print.secr` *Print secr Object*

Description

Print results from fitting a spatially explicit capture–recapture model.

Usage

```
## S3 method for class 'secr':
print (x, newdata = NULL, alpha = 0.05, deriv = FALSE, ...)
```

Arguments

`x`	`secr` object output from `secr.fit`
`newdata`	optional dataframe of values at which to evaluate model
`alpha`	alpha level
`deriv`	logical for calculation of derived D and esa
`...`	other arguments (not used currently)

Details

Results are potentially complex and depend upon the analysis (see below). Optional `newdata` should be a dataframe with a column for each of the variables in the model. If `newdata` is missing then a dataframe is constructed automatically. Default `newdata` are for a naive animal on the first occasion; numeric covariates are set to zero and factor covariates to their base (first) level. Confidence intervals are 100 (1 – alpha) % intervals.

call	the function call
time	date and time of completion
N animals	number of distinct animals detected
N captures	number of detections
N occasions	number of sampling occasions
N detectors	number of detectors
Detector type	'single', 'multi', 'proximity' etc.
Model	model formula for each 'real' parameter
Fixed	fixed real parameters
Detection fn	detection function type (halfnormal or hazard-rate)
N parameters	number of parameters estimated
Log likelihood	log likelihood
AIC	Akaike's information criterion
AICc	AIC with small sample adjustment (Burnham and Anderson 2002)
Beta parameters	coef of the fitted model, SE and confidence intervals
vcov	variance-covariance matrix of beta parameters
Real parameters	fitted (real) parameters evaluated at base levels of covariates
Derived parameters	derived estimates of density and mean effective sampling area

Derived parameters (see `derived`) are computed only for models fitted by maximizing the conditional likelihood (CL = TRUE).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Burnham, K. P. and Anderson, D. R. (2002) *Model selection and multimodel inference: a practical information-theoretic approach*. Second edition. New York: Springer-Verlag.

See Also

AIC.secr, secr.fit

Examples

```
## load & print previously fitted null (constant parameter) model
data(secrdemo)
print(secrdemo.0)
print(secrdemo.CL, deriv = TRUE)
```

---

print.traps *Print Detectors*

Description

Print method for traps objects.

Usage

```
## S3 method for class 'traps':
print(x, ...)
```

Arguments

x	traps object
...	arguments to pass to print.default

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

print, traps

Examples

```
print(make.grid(nx = 5, ny = 3))
```

rawdata *Demonstration Data*

Description

Two dataframes containing the demonstration data from program Density 4.1 (Efford 2007).

Usage

```
data(rawdata)
```

Details

Fictitious data on 235 captures of 76 individuals over 5 occasions. 100 traps 30 metres apart on a square grid with origin (365 365).

Dataframe `trapXY` contains the data from the Density input file 'trap.txt', and `captXY` contains the data from 'capt.txt'.

Source

Efford, M. G. (2007) Density 4.1: software for spatially explicit capture-recapture. Department of Zoology, University of Otago, Dunedin, New Zealand. http://www.otago.ac.nz/density.

See Also

`read.traps`, `make.grid`, `make.capthist`

Examples

```
## load demonstration data
data(rawdata)  ## trapXY, captXY
## construct a traps object from raw trap data
temptrap <- read.traps(data = trapXY, detector = 'single')
plot(temptrap)
## construct a capthist object
tempcapt <- make.capthist(captXY, temptrap, fmt='XY')
plot(tempcapt, tracks = TRUE)
## fmt='trapID' is usually simpler
```

rbind.capthist *Combine capthist Objects*

Description

Form a single `capthist` object from two or more compatible `capthist` objects.

Usage

```
rbind.capthist(..., renumber = TRUE, pool = NULL)
MS.capthist(...)
```

Arguments

...	one or more simple capthist objects (i.e., single-session)
renumber	assign new composite individual ID: sourceobject.oldID
pool	list of indices

Details

In its simplest usage, the source objects in ...each provide detection histories from a single session, and the result is a single-session object. For this to work the objects must be compatible. capthist objects are compatible if they use the same detectors (traps) and have consistent covariates and other attributes.

If the ...argument includes at least one multi-session capthist object then the elements will be formed into a single multi-session capthist object. If ...is a single multi-session object then the components of pool are used to define combinations of old sessions (e.g. pool = list(1:3, 4:5) forms two new sessions from 5 old ones).

Although rbind.capthist looks like an S3 method, it isn't. The full function name must be used.

MS.capthist treats each source object as the data for a separate session. Compatibility is not required.

Value

An object of class capthist with number of rows equal to the sum of the rows in the input objects.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

capthist, subset.capthist

Examples

```
## simulate 2-part mixture
temptrap <- make.grid(nx = 8, ny = 8)
temp1 <- sim.capthist(temptrap,
    detectpar = list(g0 = 0.1, sigma = 40))
temp2 <- sim.capthist(temptrap,
    detectpar = list(g0 = 0.2, sigma = 20))
temp3 <- rbind.capthist(temp1, temp2)

## compare mixture to sum of components
## note 'traps visited' is not additive for 'multi' detector
## nor is 'traps set'
(summary(temp1)$counts +  summary(temp2)$counts) -
    summary(temp3)$counts
```

rbind.popn *Combine popn Objects*

Description

Form a single popn object from two or more existing popn objects, or a list.

Usage

```
rbind.popn(..., renumber = TRUE)
```

Arguments

...	one or more popn objects, or a single list of popn objects
renumber	logical for whether row names in the new object should be set to the row indices

Details

An attempt to combine objects will fail if they conflict in their covariates attributes. This is not an S3 method.

Value

An object of class popn with number of rows equal to the sum of the rows in the input objects.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

popn

Examples

```
## generate and combine two subpopulations
trapobj <- make.grid()
p1 <- sim.popn(D = 3, core = trapobj)
p2 <- sim.popn(D = 2, core = trapobj)
covariates(p1) <- data.frame(size = rep('small', nrow(p1)))
covariates(p2) <- data.frame(size = rep('large', nrow(p2)))
pop <- rbind.popn(p1,p2)
```

`rbind.traps` *Combine traps Objects*

Description

Form a single `traps` object from two or more existing `traps` objects.

Usage

```
## S3 method for class 'traps':
rbind(..., renumber = TRUE)
```

Arguments

`...`	one or more `traps` objects
`renumber`	logical for whether row names in the new object should be set to the row indices

Details

An attempt to combine objects will fail if they conflict in their `usage` or `covariates` attributes.

Value

An object of class `traps` with number of rows equal to the sum of the rows in the input objects.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`, `subset.traps`

Examples

```
## nested hollow grids
hollow1 <- make.grid(nx = 8, ny = 8, hollow = TRUE)
hollow2 <- shift(make.grid(nx = 6, ny = 6, hollow = TRUE),
    c(20, 20))
nested <- rbind (hollow1, hollow2)
plot(nested, gridlines = FALSE, lab = TRUE)
```

`read.captures` *Read Capture Data From File*

Description

Input raw detection (capture) data for spatially explicit capture–recapture.

Usage

```
read.captures(file, ...)
```

Arguments

`file`	character string with name of text file
`...`	other arguments to be passed to `read.table`

Details

This brief function merely calls `read.table` and is therefore technically redundant. It accepts whatever data are in `file`. The result is used as input to `make.capthist`.

Value

A dataframe.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`make.capthist`

Examples

```
## Not run:
   ## assumes 'capt.txt' in working folder
   temp <- read.captures(file = 'capt.txt', col.names =
               c('Session','ID','Occasion', 'X', 'Y'))

## End(Not run)
```

`read.mask` *Read Habitat Mask From File*

Description

Read coordinates of points on a habitat mask from a text file.

Usage

```
read.mask(file, spacing = NULL, ...)
```

Arguments

`file`	character string with name of text file
`spacing`	spacing of grid points in metres
`...`	other arguments to pass to `read.table`

Details

Usually, the x and y coordinates are the first two values on each line, separated by white space. If the file starts with a line of column headers and 'header = TRUE' is passed to `read.table` in the ... argument then 'x' and 'y' need not be the first two fields.

If the grid cell size `spacing` is not provided then an attempt is made to infer it from the minimum spacing of points. This can be slow and may demand more memory than is available. In rare cases (highly fragmented masks) it may also yield the wrong answer.

Value

object of class `mask` with type 'user'

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`mask`

Examples

```
## Replace file name with a valid local name and remove '#'
# read.mask (file = 'c:\\myfolder\\mask.txt', spacing = 3, header = TRUE)
## 'mask.txt' should have lines like this
# x   y
# 265 265
# 268 265
# ...
```

`read.traps` *Read Detector Data From File*

Description

Construct an object of class `traps` with detector locations from a text file or data frame. Usage per occasion and covariates may be included.

Usage

```
read.traps(file = NULL, data = NULL, detector = "multi", ...)
```

Arguments

`file`	character string with name of text file
`data`	data frame of detector coordinates
`detector`	character string for detector type
`...`	other arguments to pass to `read.table`

Details

Reads a text file in which the first column is a character string identifying a detector and the next two columns are its x- and y-coordinates, separated by white space. The coordinates optionally may be followed by a string of codes '0' or '1' indicating whether the detector was operated on each occasion. A single trap-specific numeric covariate is allowed; it should be at the end of the line preceded by '/'. This format is compatible with the Density software (Efford 2007), except that all detectors are assumed to be of the same type (usage codes greater than 1 are treated as 1).

If `file` is missing then x-y coordinates will be taken instead from `data`. This option does not allow for `covariates` or `usage`, but they maybe added later.

`detector` specifies the behaviour of the detector following Efford et al. (2009). 'single' refers to a trap that is able to catch at most one animal at a time; 'multi' refers to a trap that may catch more than one animal at a time. For both 'single' and 'multi' detectors a trapped animals can appear at only one detector per occasion. Detectors of type 'proximity', such as camera traps and hair snags for DNA sampling, allow animals to be recorded at several detectors on one occasion.

Value

An object of class `traps` comprising a data frame of x- and y-coordinates, the detector type ('single', 'multi', or 'proximity'), and possibly other attributes.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G. (2007) *Density 4.1: software for spatially explicit capture–recapture*. Department of Zoology, 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 and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer, New York. Pp. 255–269.

See Also

traps, make.grid

Examples

```
## Replace file name with a valid local name and remove '#'
# read.traps ('c:\\myfolder\\mytraps.txt', detector='proximity')
## 'mytraps.txt' should have lines like this
# 1     365     365
# 2     365     395
# 3     365     425
# etc.
```

---

reduce *Combine Columns*

Description

Combine columns in a matrix-like object to create a new data set using the first non-zero value.

Usage

```
reduce (object, columns, ...)
```

Arguments

object	object that may be coerced to a matrix
columns	list in which each component is a vector of subscripts for columns to be pooled
...	other arguments (not used currently)

Details

The first element of columns defines the columns of object for the first new column, the second for the second new column etc. This is a generic method. A method exists for objects of class capthist.

Value

A matrix with number of columns equal to length(columns).

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

capthist, reduce.capthist

Examples

```
## matrix with random zeros
temp <- matrix(runif(20), nc = 4)
temp[sample(20,10)] <- 0
temp

reduce(temp, list(1:2, 3:4))
```

---

reduce.capthist *Combine Occasions*

Description

Combine columns (occasions) in a `capthist` object to create a new data set, possibly converting between detector types

Usage

```
## S3 method for class 'capthist':
reduce(object, columns = NULL, outputdetector =
    detector(traps(object)), select='last', dropunused = TRUE,
    verify = TRUE, sessions = NULL, ...)
```

Arguments

`object`	`capthist` object
`columns`	list in which each component is a vector of subscripts for occasions to be pooled
`outputdetector`	character value giving detector type for output
`select`	character value for method to resolve conflicts
`dropunused`	logical, if TRUE any never-used detectors are dropped
`verify`	logical, if TRUE the `verify` function is applied to the output
`sessions`	vector of session indices or names (optional)
`...`	other arguments (not used currently)

Details

The first component of `columns` defines the columns of `object` for new occasion 1, the second for new occasion 2, etc. If `columns` is NULL then all occasions are output. When the output detector is one of the trap types ('single', 'multi'), reducing capture occasions can result in locational ambiguity for individuals caught on more than one occasion, and for single-catch traps there may also be conflicts between individuals at the same trap. The method for resolving conflicts among 'multi' detectors is determined by `select` which should be one of 'first', 'last' or 'random'. With 'single' detectors `select` is ignored and the method is: first, randomly select* one trap per animal per day; second, randomly select* one animal per trap per day; third, when collapsing multiple days

use the first capture, if any, in each trap. With signal detectors, `select` determines how multiple signal measurements are combined; options 'min', 'max' or 'mean' are also allowed.

Usage data in the `traps` attribute are also pooled if present; a trap is considered 'used' on a pooled occasion if it was used on any contributing occasion.

* i.e., in the case of a single capture, use that capture; in the case of multiple 'competing' captures draw one at random.

Value

An object of class `capthist` with number of columns equal to `length(occasions)`. The detector type is inherited from `object` unless a new type is specified with the argument `outputdetector`.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`capthist`, `subset.capthist`

Examples

```
tempcapt <- sim.capthist (make.grid(nx = 6, ny = 6), nocc = 6)
class(tempcapt)

pooled.tempcapt <- reduce(tempcapt, col = list(1,2:3,4:6))
summary (pooled.tempcapt)
```

---

rotate *Rotate Points*

Description

Rotate a set of points.

Usage

```
rotate (object, degrees, centrexy = NULL, ...)
```

Arguments

`object`	object that may be coerced to a numeric matrix
`degrees`	clockwise angle of rotation in degrees
`centrexy`	vector with xy coordinates of rotation centre
`...`	other arguments (not used)

Details

The first column of `object` holds the x coordinates of the points and the second holds the y coordinates. If `centrexy` is NULL then rotation is about (0,0).

A generic method, introduced for the class-specific method `rotate.traps`.

Value

A matrix with the location of each point rotated about the centre.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

shift

Examples

```
temp <- matrix(runif (20) * 2 - 1, nc = 2)
temp2 <- rotate(temp, 25)
plot(temp, xlim=c(-1.5,1.5), ylim = c(-1.5,1.5), pch = 16)
points (0,0, pch=2)
points (temp2, pch = 1)
arrows (temp[,1], temp[,2], temp2[,1], temp2[,2], length = 0.1)
```

---

rotate.traps *Rotate Detectors*

Description

Rotate detectors while retaining other attributes.

Usage

```
## S3 method for class 'traps':
rotate(object, degrees, centrexy = NULL, ...)
```

Arguments

object	object of class traps
degrees	clockwise angle of rotation in degrees
centrexy	vector with x,y coordinates of point about which to rotate
...	other arguments (not used).

Details

May be used with rbind.traps and shift.traps to create complex geometries.

Value

An object of class traps with the location of each detector rotated about the centre.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

traps, shift.traps

Examples

```
hollow1 <- make.grid(nx = 8, ny = 8, hollow = TRUE)
hollow2 <- make.grid(nx = 8, ny = 8, hollow = TRUE)
nested <- rbind (hollow1, rotate(hollow2, 45, c(70, 70)))
plot(nested, gridlines = FALSE)
```

---

score.test *Score Test for SECR Models*

Description

Compute score tests comparing a fitted model and a more general alternative model.

Usage

```
score.test(secr, ..., betaindex = NULL, trace = FALSE)

score.table(object, ..., sort = TRUE, dmax = 10)
```

Arguments

secr	fitted secr model
...	one or more alternative models OR a fitted secr model
trace	logical. If TRUE then output one-line summary at each evaluation of the likelihood
betaindex	vector of indices mapping fitted values to parameters in the alternative model
object	score.test object or list of such objects
sort	logical for whether output rows should be in descending order of AICc
dmax	threshold of dAICc for inclusion in model set

Details

Score tests allow fast model selection (e.g. Catchpole & Morgan 1996). Only the simpler model need be fitted. This implementation uses the observed information matrix, which may sometimes mislead (Morgan et al. 2007). The gradient and second derivative of the likelihood function are evaluated numerically at the point in the parameter space of the second model corresponding to the fit of the first model. This operation uses the function fdHess of the **nlme** package; the likelihood must be evaluated several times, but many fewer times than would be needed to fit the model. The score statistic is an approximation to the likelihood ratio; this allows the difference in AIC to be estimated.

Mapping of parameters between the fitted and alternative models sometimes requires user intervention via the betaindex argument. For example betaindex = c(1,2,4) is the correct mapping when comparing the null model (D$\sim$ 1, g0$\sim$ 1, sigma$\sim$ 1) to one with a behavioural effect on g0 ((D$\sim$ 1, g0$\sim$ b, sigma$\sim$ 1).

score.table summarises one or more score tests in the form of a model comparison table. The ... argument here allows the inclusion of additional score test objects (note the meaning differs from score.test). Approximate AICc values are used to compute relative AIC model weights for all models within dmax AICc units of the best model.

Value

An object of class 'score.test' that inherits from 'htest', a list with components

statistic	the value the chi-squared test statistic (score statistic)
parameter	degrees of freedom of the approximate chi-squared distribution of the test statistic (difference in number of parameters H0, H1)
p.value	probability of test statistic assuming chi-square distribution
method	a character string indicating the type of test performed
data.name	character string with null hypothesis, alternative hypothesis and arguments to function call from fit of H0
H0	simpler model
np0	number of parameters in simpler model
H1	alternative model
H1.beta	coefficients of alternative model
AIC	Akaike's information criterion, approximated from score statistic
AICc	AIC with small-sample adjustment of Hurvich & Tsai 1989

If ... defines several alternative models then a list of score.test objects is returned.

The output from score.table is a dataframe with one row per model, including the reference model.

Note

This implementation is experimental. The AIC values, and values derived from them, are approximations that may differ considerably from AIC values obtained by fitting and comparing the respective models. Use of the observed information matrix may not be optimal.

score.test cannot be used to compare models that differ in the arguments scalesigma or scaleg0.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Catchpole, E. A. and Morgan, B. J. T. (1996) Model selection of ring-recovery models using score tests. *Biometrics* **52**, 664–672.

Hurvich, C. M. and Tsai, C. L. (1989) Regression and time series model selection in small samples. *Biometrika* **76**, 297–307.

Morgan, B. J. T., Palmer, K. J. and Ridout, M. S. (2007) Negative score test statistic. *American statistician* **61**, 285–288.

See Also

AIC, LR.test

Examples

```
## Not run:
    data(secrdemo)
    AIC (secrdemo.0, secrdemo.b)
    st <- score.test (secrdemo.0, g0 ~ b)
    st
    score.table(st)

## End(Not run)
```

---

secr.design.MS *Construct Detection Model Design Matrices and Lookups*

Description

Internal function used by secr.fit.

Usage

```
secr.design.MS(capthist, models, timecov = NULL, sessioncov = NULL,
    groups = NULL, dframe = NULL, naive = FALSE, bygroup = FALSE, ...)
```

Arguments

capthist	capthist object
models	list of formulae for parameters of detection
timecov	optional dataframe of values of time (occasion-specific) covariate(s).
sessioncov	optional dataframe of values of session-specific covariate(s).
groups	optional vector of one or more variables with which to form groups. Each element should be the name of a factor variable in the covariates attribute of capthist.
dframe	optional data frame of design data for detection parameters
naive	logical if TRUE then modelled detection probability is for a naive animal (not caught previously); if FALSE then detection probability is contingent on individual's history of detection
bygroup	logical if TRUE then the individual dimension of the parameter matrix is automatically collapsed to one row per group; if FALSE then the full dimensionality is retained (one row per individual)
...	other arguments passed to the R function model.matrix

Details

This is an internal **secr** function that you are unlikely ever to use. ... may be used to pass contrasts.arg to model.matrix.

Each real parameter is notionally different for each unique combination of session, individual, occasion and detector, i.e., for R sessions, n individuals, S occasions and K detectors there are *potentially* $R \times n \times S \times K$ different values. Actual models always predict a much reduced set of distinct values, and the number of rows in the design matrix is reduced correspondingly; a parameter index array allows these to retrieved for any combination of session, individual, occasion and detector.

Value

A list with the components

`designMatrices`
: list of reduced design matrices, one for each real detection parameter

`parameterTable`
: index to row of the reduced design matrix for each real detection parameter; dim(parameterTable) = c(uniquepar, np), where uniquepar is the number of unique combinations of paramater values (uniquepar < $RnSK$) and np is the number of parameters in the detection model.

`PIA`
: Parameter Index Array - index to row of parameterTable for a given session, animal, occasion and detector; dim(PIA) = c(R,n,S,K)

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

D.designdata

Examples

```
data(captdata)
secr.design.MS (captdata, models = list(g0 = ~b))$designMatrices
secr.design.MS (captdata, models = list(g0 = ~b))$parameterTable
```

---

secr.fit *Spatially Explicit Capture–Recapture*

Description

Estimate animal population density with data from an array of passive detectors (traps) by fitting a spatial detection model by maximizing the likelihood. Data must have been assembled as an object of class `capthist`. Integration is by summation over the grid of points in `mask`.

Usage

```
secr.fit (capthist, model = list(D~1, g0~1, sigma~1),
    mask = NULL, buffer = 100, CL = FALSE, detectfn = NULL,
    start = NULL, link = list(), fixed = list(),
    timecov = NULL, sessioncov = NULL, groups = NULL, dframe = NULL,
    details = list(), method = 'Newton-Raphson', verify = TRUE, trace = NULL, ..
```

Arguments

`capthist`	`capthist` object including capture data and detector (trap) layout
`mask`	`mask` object
`buffer`	scalar mask buffer radius if `mask` not specified
`CL`	logical, if true then the model is fitted by maximizing the conditional likelihood

detectfn	integer code for shape of detection function. 0 halfnormal, 1 hazard-rate, 2 exponential etc.
start	vector of initial values for beta parameters, or secr object from which they may be derived
link	list with optional components 'D', 'g0', 'sigma' and 'z', each a character string in {'log', 'logit', 'identity', 'sin'} for the link function of the relevant real parameter
fixed	list with optional components corresponding to each 'real' parameter (e.g., 'D', 'g0', 'sigma'), the scalar value to which parameter is to be fixed
model	list with optional components 'D', 'g0', 'sigma' and 'z', each symbolically defining a linear predictor for the relevant real parameter using formula notation
timecov	optional dataframe of values of time (occasion-specific) covariate(s).
sessioncov	optional dataframe of values of session-specific covariate(s).
groups	optional vector of one or more variables with which to form groups. Each element should be the name of a factor variable in the covariates attribute of capthist.
dframe	optional data frame of design data for detection parameters
details	list of additional settings, mostly model-specific (see Details)
method	character string giving method for maximizing log likelihood
verify	logical, if TRUE the input data are checked with verify
trace	logical, if TRUE then output each evaluation of the likelihood, and other messages
...	other arguments passed to the maximization function

Details

secr.fit fits a SECR model by maximizing the likelihood. The likelihood depends on the detector type ('multi' or 'proximity') of the 'traps' attribute of capthist (Borchers and Efford 2008, Efford, Borchers and Byrom 2009, Efford, Dawson and Borchers 2009). The 'multi' form of the likelihood is also used, with a warning, when detector type = 'single' (see Efford et al. 2009). Default model is null (constant density and detection probability). The set of variables available for use in linear predictors includes some that are constructed automatically (t, T, b, B), group (g), and others that appear in the covariates of the input data. See secr models for more on defining models.

The length of timecov should equal the number of sampling occasions (ncol(capthist)). Arguments timecov, sessioncov and groups are used only when needed for terms in one of the model specifications. Default link is list(D='log', g0='logit', sigma='log').

If start is missing then autoini is used for D, g0 and sigma, and other beta parameters are set initially to arbitrary values, mostly zero. start may be a previously fitted nested model. In this case, a vector of starting beta values is constructed from the nested model and additional betas are set to zero. Mapping of parameters follows the default in score.test, but user intervention is not allowed.

details is used for various specialized settings –

details$distribution specifies the distribution of the number of individuals detected; this may be conditional on the number in the masked area ('binomial') or unconditional ('poisson'). distribution affects the sampling variance of the estimated density. The default is 'poisson'.

details$hessian is a character string controlling the computation of the Hessian matrix from which variances and covariances are obtained. Options are 'none' (no variances), 'auto' (the default) or 'fdhess' (use the function fdHess in **nlme**). If 'auto' then the Hessian from the optimisation function is used.

details$LLonly = TRUE causes the function to returns a single evaluation of the log likelihood at the 'start' values.

details$scalesigma = TRUE causes sigma to be scaled by $D^{-0.5}$.

details$scaleg0 = TRUE causes g0 to be scaled by sigma^{-2}. The corresponding 'real' parameters are marked with an asterisk in output (e.g. g0*).

If method = 'Newton-Raphson' then nlm is used to maximize the log likelihood; otherwise optim is used with the chosen method ('BFGS','Nelder-Mead', etc.). A feature of nlm is that it takes a large step early on in the maximization that may cause floating point underflow or overflow in one or more real parameters. This can be controlled by passing the 'stepmax' argument of nlm in the ... argument of secr.fit (see first example).

If verify = TRUE then verify is called to check capthist and mask; analysis is aborted if errors are found.

Value

The function secr.fit returns an object of class secr. This has components

call	function call
capthist	saved input
mask	saved input
detectfn	saved input
CL	saved input
timecov	saved input
sessioncov	saved input
groups	saved input
dframe	saved input
design	reduced design matrices, parameter table and parameter index array for actual animals (see secr.design.MS)
design0	reduced design matrices, parameter table and parameter index array for 'naive' animal (see secr.design.MS)
start	vector of starting values for beta parameters
link	list with components for each real parameter (e.g., 'D', 'g0'), the name of the link function used for each real parameter. Component 'z' is NULL unless detectfn = 1 (hazard-rate).
fixed	saved input
parindx	list with possible components 'D', 'g0', 'sigma' and 'z', for the indices of the 'beta' parameters associated with each real parameter ('z' NULL unless detectfn = 1).
model	saved input
details	saved input
vars	vector of unique variable names in model
betanames	names of beta parameters

realnames	names of fitted (real) parameters
fit	list describing the fit (output from nlm or optim)
beta.vcv	variance-covariance matrix of beta parameters
D	array of predicted densities of each group at each mask point in each session, dim(D) = c(nrow(mask), ngroups, nsessions)
version	secr version number
starttime	character string of date and time at start of fit
proctime	processor time for model fit, in seconds

Note

print, AIC, vcov, and predict methods are provided. derived is used to compute the derived parameters 'esa' (effective sampling area) and 'D' (density) for models fitted by maximizing the conditional likelihood (CL = TRUE).

Components 'version' and 'starttime' were introduced in version 1.2.7, and recording of the completion time in 'fitted' was discontinued.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

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

Efford, M. G., Borchers D. L. and Byrom, A. E. (2009) Density estimation by spatially explicit capture–recapture: likelihood-based methods. In: D. L. Thompson, E. G. Cooch and 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.

See Also

capthist, mask, detection functions, print.secr, vcov.secr, AIC.secr, derived, predict.secr, verify

Examples

```
## construct test data (array of 48 'multi-catch' traps)

detectors <- make.grid (nx = 6, ny = 8, detector = 'multi')
detections <- sim.capthist (detectors, popn = list(D = 10,
    buffer = 100), detectpar = list(g0 = 0.2, sigma = 25))

## fit & print null (constant parameter) model
## stepmax is passed to nlm (not usually needed)

secr0 <- secr.fit (detections, stepmax = 50)
secr0   ## uses print method for secr
```

```
## compare fit of null model with learned-response model for g0

secrb <- secr.fit (detections, model = g0~b)
AIC (secr0, secrb)

## typical result

##                  model   detectfn npar    logLik     AIC    AICc dAICc  AICwt
## secr0 D~1 g0~1 sigma~1 halfnormal    3 -347.1210 700.242 700.928 0.000 0.7733
## secrb D~1 g0~b sigma~1 halfnormal    4 -347.1026 702.205 703.382 2.454 0.2267
```

`secr.make.newdata` *Create Default Design Data*

Description

Generate a dataframe containing design data for the base levels of all predictors in an secr object.

Usage

```
secr.make.newdata(object)
```

Arguments

`object`	fitted secr model object

Details

`secr.make.newdata` is used by `predict` in lieu of user-specified 'newdata'. There is seldom any need to call `secr.make.newdata` directly.

Value

A dataframe with one row for each session and group, and columns for the predictors used by `object$model`.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`predict.secr`, `secr.fit`

Examples

```
data(secrdemo)
secr.make.newdata(secrdemo.b)
```

`secr.model` *Spatially Explicit Capture–Recapture Models*

Description

A family of capture–recapture models (e.g. SECR) may include submodels that constrain variation in core parameters and include the effects of covariates. 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 effects 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, +Inf).

Submodels for spatially explicit capture–recapture in **secr** are defined symbolically using the 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 to reduce confusion. Four real parameters are commonly modelled in **secr** 1.3: D (density), g0, sigma and z. Only the last three real parameters, the ones jointly defining detection probability as a function of location, can be estimated directly when the model is fitted by maximizing the conditional likelihood. D is then a derived parameter. 'z' is a shape parameter used only for a 'hazard-rate' detection function (Hayes and Buckland 1983). Other real parameters are used for acoustic models (beta0, beta1; `../doc/secr-sound.pdf`) and for the mixture proportion (pmix) in finite mixture models (`../doc/secr-finitemixtures.pdf`).

Each real parameter has a linear predictor of the form

y = X * beta,

where y is vector of parameter values on the link scale, X is a design matrix of predictor values, beta is a vector of coefficients, and '*' stands for matrix multiplication. The elements of beta are estimated when we fit the model; in MARK these are called 'beta parameters' to distinguish them from the 'real' parameter values in y. X has one column for each element of beta. To repeat: there is an X and a beta for each real parameter; elsewhere in the documentation we use 'beta' to refer to the vector got by concatenating *all* the parameter-specific beta's. We now describe design matrices in more detail.

[Some variations on the basic SECR model do not fit easily into this framework. An example is the choice of detection function (halfnormal vs hazard-rate). These are treated as higher-level choices.]

Design matrices

The design matrix contains a column of '1's (for the constant or intercept term) and additional columns as needed to describe the effects in the submodel. 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`, a design matrix is constructed automatically from the input data (`capthist`) and the model formula (e.g. `model$g0`) 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, and more if the formula includes interactions or factors with more than two levels. For a good description of the general approach see the documentation for RMark (Laake and Rexstad 2008). The key point is that the necessary design data can be either extracted from the inputs (`capthist` and `mask`) or generated automatically (e.g. indicator of previous capture, mentioned in the previous paragraph).

Real parameters fall into two groups: density (D) and detection (g0, sigma and z). Density and detection parameters are subject to different types of effect, so they use different design matrices and are described separately here:

`secr detection models`, `secr density models`

Note

The structure of **secr** precludes certain types of model. Unlike density, detection parameters (g0, sigma etc.) cannot be modelled as varying in space *per se*, whether continuously or discretely (e.g. as a function of habitat class). However, such variation may be modelled between detectors or between sessions. As an example, consider a measure of vegetation cover in a 50-m circle centred on each detector. This may be used as a detector covariate in models for g0 or sigma. A 'detector-centred' view of habitat effects is almost as sensible as an 'animal-centred' view; the one reservation is that the spatial scale (radius of the circle) is arbitrary rather than being driven by sigma as you might like. Perhaps this could be fixed in future versions by computing the trap covariate 'on the fly' from covariates in the habitat mask, given the current magnitude of sigma.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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 www.phidot.org.

`secr.model.density` *Density Models*

Description

SECR can fit an inhomogeneous Poisson model to describe the distribution of animals. This may be viewed as a surface of expected density across the study area.

The log likelihood is evaluated in `secr.fit` by summing values at points on a 'habitat mask'. Each point in a habitat mask represents a grid cell of potentially occupied habitat (their combined area may be almost any shape and may include disjunct patches). The full design matrix for density (D) has one row for each point in the mask. 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', 'y', 'session' and 'g' are provided automatically. Other covariates should be named columns in the 'covariates' attribute of the habitat mask.

Variable	Description	Data source
x	x-coordinate	automatic
y	y-coordinate	automatic
session	session factor	automatic
g	group factor	automatic
[user]	mask covariate	`covariates(mask)` as named in formula

The submodel for density (D) is a named component of the list used in the `model` argument of `secr.fit`. It is expressed in R formula notation by appending terms to ~ .

Note

This implementation is still experimental. Note that no density model is fitted when `secr.fit` is called with CL = TRUE.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

See Also

`secr models`, `secr detection models`, `secr.fit`

Examples

```
list(D = ~ 1)     ## constant density (homogeneous Poisson)
list(D = ~ x)     ## east-west trend
list(D = ~ cover) ## requires 'cover' as a mask covariate
```

---

secr.model.detection

Models for Detection Parameters

Description

For spatially explicit capture–recapture estimation of a closed population, we model the detection of individual i on occasion s at detector k. Given n observed individuals on S occasions at K detectors there are therefore $n.S.K$ detection probabilities of interest. We can think of these as elements of 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 consider them in the array.

In a null (constant) model, all $n.S.K$ detection probabilities are the same. The conventional sources of variation in capture probability (Otis et al. 1978) appear as variation in the n dimension ('individual heterogeneity' h), 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.

Spatially explicit capture–recapture introduces two sorts of additional complexity. Firstly, detection probability is no longer a scalar (even for a particular animal, occasion and detector combination); it is described by the detection function, which may have two parameters (e.g. g0, sigma for half-normal), three parameters (e.g. g0, sigma, z for the hazard-rate function), or potentially more.

Secondly, many more types of variation are possible. Any of the parameters of the detection function may vary with respect to individual (i), occasion (s) or detector (k). For example, there may be a covariate associated with trap location that influences detection probability.

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

Variation between 'sessions' adds a further level of complexity: in principle there is an $n.S.K$ array for each session (sessions are numbered $1..R$). We do not expand on this here.

Specifying effects on detection parameters

Effects on parameters of detection probability are specified with R formulae using standard variable names or named covariates supplied by the user. The formula for each detection parameter (g0, sigma, z) may be constant ($\sim$ 1, the default) or some combination of terms in standard R formula notation (see `formula`).

Variable	Description	Data source	Dim
t	time factor (one level for each occasion)	automatic	S
T	time trend (integer covariate 0:(S-1))	automatic	S
tcov	default time covariate	timecov[,1]	S
kcov	default trap covariate	covariates (traps)[,1]	K
b	learned response	`capthist`	$n.S$
B	transient (Markovian) response	`capthist`	$n.S$
g	group	see below	n
h2	2-class mixture	–	2
h3	3-class mixture	–	2
session	session factor (one level for each session)	automatic	R
[user]	individual covariate	`covariates (capthist)`	n
[user]	session covariate	`sessioncov`	R
[user]	time covariate	`timecov`	S
[user]	detector covariate	`covariates (traps)`	K

The classic 'learned response' is a step change following first detection; this is implemented with the predictor variable 'b' which is FALSE up to and including the time of first capture and TRUE afterwards. An alternative is a response that depends only on detection at the last opportunity ('B').

Groups ('g') are defined by the interaction of the `capthist` categorical (factor) individual covariates identified in `secr.fit` argument 'groups'. Groups are redundant with conditional likelihood because individual covariates of whatever sort (continuous or categorical) may be included freely in the model.

Individual heterogeneity ('h' in the notation of Otis et al. 1978) may modelled by treating any detection parameter as a 2-part or 3-part finite mixture e.g. g0 $\sim$ h2. See `../doc/secr-finitemixtures.pdf`.

Any other variable name appearing in a formula is assumed to refer to a user-defined predictor. These will be interpreted by searching for name matches in the dataframes of individual, session, time and trap covariates, in that order (remembering that individual covariates other than groups are allowed only when the model is fitted by maximizing the conditional likelihood). The type of the predictor is inferred from the data frame in which it first occurs. Thus if the model included the formula 'g0 ~ wetness', and 'wetness' was a column in the data frame of time covariates (timecov), then 'wetness' would be interpreted as a time covariate, and a column of the same name in covariates(traps) would be ignored. In this case, renaming the column in timecov would expose the

traps covariate, and 'wetness' would be interpreted as an attribute of detectors, rather than sample intervals. This is a good reason to give covariates distinctive names!

The design matrix for detection parameters may also be provided manually in the argument `dframe`. This feature is untested.

The submodels for 'g0', 'sigma' and 'z' are named components of the `model` argument of `secr.fit`. They are expressed in R formula notation by appending terms to ~ . The name of the response may optionally appear on the left hand side of the formula (e.g. g0~ b).

Note

The parameter 'z' was previously called 'b'; it was renamed to avoid confusion with the predictor b used in a formula for a learned trap response.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

Hayes, R. J. and Buckland, S. T. (1983) Radial-distance models for the line-transect method. *Biometrics* **39**, 29–42.

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

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

See Also

secr models, secr density models, secr.fit

Examples

```
## constant (null) model
list(g0 = ~1, sigma = ~1)

## both detection parameters change after first capture
list(g0 = ~b, sigma = ~b)

## group-specific parameters; additive time effect on g0
## groups are defined via the 'groups' argument of secr.fit
list(g0 = ~ g + t, sigma = ~ g)

## g0 depends on trap-specific covariate
list(g0 = ~ kcov)
```

secrdemo *SECR Models Fitted to Demonstration Data*

Description

Objects of class secr formed by

```
secrdemo.0 <- secr.fit (captdata)
```

```
secrdemo.b <- secr.fit (captdata, model = list(g0 ~b))
```

```
secrdemo.CL <- secr.fit (captdata, CL = TRUE)
```

where captdata is a capthist object using the demonstration data (files 'trap.txt','capt.txt') from program Density 4.1 (Efford 2007).

Usage

```
data(secrdemo)
```

Details

The raw data are 235 fictional captures of 76 animals in 100 single-catch traps over 5 occasions. The fitted models use a halfnormal detection function and the likelihood for multi-catch traps (expect estimates of g0 to be biased because of trap saturation Efford et al. 2009). The first is a null model (i.e. parameters constant) and the second fits a learned trap response.

Source

Efford, M.G. (2007) Density 4.1: software for spatially explicit capture-recapture. Department of Zoology, 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 and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer, New York. Pp. 255–269.

See Also

captdata, capthist

Examples

```
## display the default fit with the print method for secr
data (secrdemo)
secrdemo.0
```

`session` *Session Vector*

Description

Extract or replace the session names of a `capthist` object.

Usage

```
session(object, ...)
session(object) <- value
```

Arguments

`object`	object with 'session' attribute e.g. `capthist`
`value`	character vector or vector that may be coerced to character, one value per session
`...`	other arguments (not used)

Details

Replacement values will be coerced to character.

Value

a character vector with one value for each session in `capthist`

Note

Like Density, **secr** uses the term 'session' for a closed-population sample. A session usually includes data from several closely-spaced capture occasions (often consecutive days). Each 'primary session' in the 'robust' design of Pollock (1982) would be treated as a session in **secr**. **secr** also uses 'session' for independent subsets of the capture data distinguished by characteristics other than sampling time (as above). For example, two grids trapped simultaneously could be analysed as distinct 'sessions' if (i) they were far enough apart that there was negligible prospect of the same animal being caught on both grids, and (ii) there was interest in comparing estimates from the two grids, or fitting a common detection model.

The log likelihood for a session model is the sum of the separate session log likelihoods. Although this assumes independence of sampling, parameters may be shared across sessions, or session-specific parameter values may be functions of session-level covariates. For many purposes, 'sessions' are equivalent to 'groups'. For multi-session models the detector array and mask are specified separately for each session. Group models are therefore generally simpler to implement. On the other hand, sessions offer more flexibility in defining and evaluating between-session models, including trend models.

Sessions are a recent addition to **secr** and the documentation and testing of session capability is therefore less advanced than for other features.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Pollock, K. H. (1982) A capture-recapture design robust to unequal probability of capture. *Journal of Wildlife Management* **46**, 752–757.

See Also

`capthist`

Examples

```
data(captdata)
session(captdata)
```

`setnullsignal` *Null signal strength*

Description

Set the signal value to be used for non-detections.

Usage

```
setnullsignal (value)
```

Arguments

`value`	numeric value

Details

A single numeric value is used internally to represent signal strength below the threshold of detection. The value should be less than any valid signal measurement.

The default value is 0 (zero). This is adequate for most terrestrial acoustic applications, in which signal strength is typically in the range 40 to 120 dB.

Value

The old null value is returned. A side effect is to change the 'nullval' variable in the namespace of the package.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
setnullsignal(-1e20)
```

`shift` *Shift Points*

Description

Translate an array of points.

Usage

```
shift (object, shiftxy, ...)
```

Arguments

`object`	a 2-column matrix or object that can be coerced to a matrix
`shiftxy`	vector of x and y displacements
`...`	other arguments (not used)

Details

This is a generic function. The default method is redundant, but the method for `traps` objects may be useful.

Value

A matrix with the location of each point shifted by the desired amount.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`rotate`, `flip`

Examples

```
temp <- matrix(runif (20) * 2 - 1, nc = 2)
temp2 <- shift(temp, c(0.1, 0.1))
plot(temp, xlim=c(-1.5,1.5), ylim = c(-1.5,1.5), pch = 16)
points (0,0, pch=2)
points (temp2, pch = 1)
arrows (temp[,1], temp[,2], temp2[,1], temp2[,2], length = 0.1)
```

shift.traps *Shift Detectors*

Description

Translate detectors while retaining other attributes.

Usage

```
## S3 method for class 'traps':
shift(object, shiftxy, ...)
```

Arguments

object	object of class traps
shiftxy	vector with displacements in x and y directions
...	other arguments (not used)

Details

May be used with rbind.traps and rotate.traps to create complex geometries.

Value

An object of class traps with the location of each detector shifted by the desired amount.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

traps, rotate.traps, flip.traps

Examples

```
hollow1 <- make.grid(nx = 8, ny = 8, hollow = TRUE)
hollow2 <- shift(make.grid(nx = 6, ny = 6, hollow = TRUE), c(20, 20))
nested <- rbind (hollow1, hollow2)
plot(nested, gridlines = FALSE, lab = TRUE)
```

`sim.capthist` *Simulate Detection Histories*

Description

Create a set of capture or marking-and-resighting histories by simulated sampling of a 2-D population using an array of detectors.

Usage

```
sim.capthist(traps, popn = list(D = 5, buffer = 100,
    Ndist = 'poisson'), detectfn = 0, detectpar = list(),
    noccasions = 5, renumber = TRUE, seed = NULL)
sim.resight(traps, ..., q = 1, pID = 1, unmarked = TRUE,
    nonID = TRUE)
```

Arguments

`traps`	`traps` object with the locations and other attributes of detectors
`popn`	locations of individuals in the population to be sampled, either as a `popn` object or a list with named elements 'D' (density) and 'buffer'
`detectfn`	code for type of detection function
`detectpar`	list of values for named parameters of detection function
`noccasions`	number of occasions to simulate
`renumber`	logical for whether output rows should labeled sequentially (TRUE) or retain the numbering of the population from which they were drawn (FALSE)
`seed`	an object specifying if and how the random number generator should be initialized ('seeded')
`...`	arguments to pass to `sim.capthist`
`q`	number of marking occasions
`pID`	probability of individual identification for marked animals
`unmarked`	logical, if true unmarked individuals are not recorded during 'sighting'
`nonID`	logical, if true then unidentified marked individuals are not recorded during 'sighting'

Details

If `popn` is not of class 'popn' then a homogeneous Poisson population with the desired density (animals/ha) is first simulated over the rectangular area of the bounding box of `traps` plus a buffer of the requested width (metres). The detection algorithm depends on the detector type of `traps`. For 'proximity' detectors, the actual detection probability of animal i at detector j is the naive probability given by the detection function. For 'single' and 'multi' detectors the naive probability is modified by competition between detectors and, in the case of 'single' detectors, between animals. See Efford (2004) and other papers below for details.

Detection parameters are specific to the detection function, which is indicated by a numeric code (`detection functions`). Parameters may vary with time - for this provide a vector of length

noccasion. The default detection parameters are list(g0 = 0.2, sigma = 25, z = 1).

detectpar may also include 'binomN' and other arguments for detectors that have yet to be documented. A zero value for binomN indicates that counts should be modelled with a Poisson distribution.

If popn is specified by an object of class 'popn' then any individual covariates will be passed on; the covariates attribute of the output is otherwise set to NULL.

The random number seed is managed as in simulate.

sim.resight generates mark-resight data for 'q' marking occasions followed by 'noccasion – q' sighting occasions. sim.capthist is first called with the arguments 'traps' and .... The detector type must be 'proximity'. The 'usage' attribute of traps is ignored at present, so the same detectors are operated on all occasions. Any detection-parameter vector of length 2 in ... is interpreted as providing differing constant values for the marking and sighting phases.

Value

For sim.capthist, an object of class capthist, a matrix or 3-dimensional array with additional attributes. Rows represent individuals and columns represent occasions; the third dimension, used when detector type = 'proximity', codes presence or absence at each detector. For trap detectors ('single', 'multi') each entry in capthist is either zero (no detection) or the sequence number of the trap.

The initial state of the R random number generator is stored in the 'seed' attribute.

For sim.resight, an object of class capthist, always a 3-dimensional array, with additional attributes Tu and Tm containing counts of 'unmarked' and 'marked, not identified' sightings.

Note

External code is called to speed the simulations. The present version assumes a 'null model' i.e. naive detection probability is constant except for effects of distance and possibly time (using vector-valued detection parameters from 1.2.10). You can, however, use rbind.capthist to combine detections of population subclasses (e.g. males and females) simulated with different parameter values. This is not valid for detector type 'single' because it fails to allow for competition for traps between subclasses. Future versions may allow more complex models.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

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

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

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

See Also

sim.popn, capthist, traps, popn, detection functions, simulate

Examples

```
## simple example
## detector = 'multi' (default)
temptrap <- make.grid(nx = 6, ny = 6, spacing = 20)
sim.capthist (temptrap, detectpar = list(g0 = 0.2, sigma = 20))

## with detector = 'proximity, there may be more than one
## detection per individual per occasion
temptrap <- make.grid(nx = 6, ny = 6, spacing = 20, detector =
    'proximity')
summary(sim.capthist (temptrap, detectpar =
    list(g0 = 0.2, sigma = 20)))
```

---

sim.popn *Simulate 2-D Population*

Description

Simulate a Poisson process representing the locations of individual animals.

Usage

```
sim.popn (D, core, buffer = 100, model2D = 'poisson',
    buffertype = 'rect', covariates = list(sex = c(M=0.5, F=0.5)),
    Ndist = 'poisson', details = NULL, seed = NULL)
```

Arguments

D	density animals / hectare (10 000 m\^2)
core	data frame of points defining the core area
buffer	buffer radius about core area
model2D	character string for 2-D distribution ('poisson', 'cluster', 'IHP')
buffertype	character string for buffer type
covariates	list of named covariates
Ndist	character string for distribution of number of individuals
details	optional list with additional parameters
seed	value for setting .Random.seed - either NULL or an integer

Details

`core` must contain columns 'x' and 'y'; a `traps` object is suitable. For buffertype = 'rect', animals are simulated in the rectangular area obtained by extending the bounding box of `core` by `buffer` metres to top and bottom, left and right. This box has area A.

A notional covariate 'sex' is generated by default.

Each element of `covariates` defines a categorical (factor) covariate with the given probabilities of membership in each class. No mechanism is provided for generating continuous covariates, but these may be added later (see Examples).

Ndist may be 'poisson' or 'fixed'. The number of individuals N has expected value DA. If DA is non-integer then Ndist = 'fixed' results in $N \in \{\mathrm{trunc}(DA), \mathrm{trunc}(DA) + 1\}$, with probabilities set to yield DA individuals on average.

If model2D = 'cluster' then the simulated population approximates a Neyman-Scott clustered Poisson distribution. Ancillary parameters are passed as components of `details`: details\$mu is the fixed number of individuals per cluster and details\$hsigma is the spatial scale (σ) of a 2-D kernel for location within each cluster. The algorithm is

1. Determine the number of clusters (parents) as a random Poisson variate with $\lambda = DA/\mu$
2. Locate each parent by drawing uniform random x- and y-coordinates
3. Generate mu offspring for each parent and locate them by adding random normal error to each parent coordinate
4. Apply toroidal wrapping to ensure all offspring locations are inside the buffered area

Toroidal wrapping is a compromise. The result is more faithful to the Neyman-Scott distribution if the buffer is large enough that only a small proportion of the points are wrapped.

If model2D = 'IHP' then an inhomogeneous Poisson distribution is fitted. `core` should be a habitat mask and `D` should be a vector of length equal to the number of cells (rows) in `core`. The number of individuals in each cell is Poisson-distributed with mean DA where A is the cell area (an attribute of the mask).

The random number seed is managed as in `simulate.lm`.

Value

An object of class 'popn', a data frame with columns 'x' and 'y'. Rows correspond to individuals. Individual covariates (optional) are stored as a data frame attribute. The initial state of the R random number generator is stored in the 'seed' attribute.

Note

Other `buffertypes` will be defined later. (e.g. convex hull, concave)

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`popn`, `simulate`

Examples

```
temppop <- sim.popn (D = 10, expand.grid(x = c(0,100), y =
    c(0,100)), buffer = 50)

## plot, distinguishing 'M' and 'F'
plot(temppop, pch = 1, cex= 1.5,
    col = c('green','red')[covariates(temppop)$sex])

## add a continuous covariate
## assumes covariates(temppop) is non-null
covariates(temppop)$size <- rnorm (nrow(temppop), mean = 15, sd = 3)
summary(covariates(temppop))

## Neyman-Scott cluster distribution
oldpar <- par(xpd = TRUE, mfrow=c(2,3))
for (h in c(5,15))
for (m in c(1,4,16)) {
    temppop <- sim.popn (D = 10, expand.grid(x = c(0,100),
        y = c(0,100)), model2D = 'cluster', buffer = 100,
        details = list(mu = m, hsigma = h))
    plot(temppop)
    text (50,230,paste(' mu =',m, 'hsigma =',h))
}
par(oldpar)

## Inhomogeneous Poisson distribution
data(secrdemo)
xy <- secrdemo.0$mask$x + secrdemo.0$mask$y - 900
tempD <- xy^2 / 1000
plot(sim.popn(tempD, secrdemo.0$mask, model2D = 'IHP'))
```

`sim.secr` *Simulate From Fitted secr Model*

Description

Simulate a spatially distributed population, sample from that population with an array of detectors, and optionally fit an SECR model to the simulated data.

Usage

```
## S3 method for class 'secr':
simulate(object, nsim = 1, seed = NULL, chat = 1, ...)

sim.secr(object, nsim = 1, extractfn = function(x) c(deviance =
    deviance(x), df = df.residual(x)), seed = NULL, data = NULL,
    tracelevel = 1, hessian = "none", start = object$fit$par)
```

Arguments

object	an secr object
nsim	number of replicates
seed	value for setting .Random.seed - either NULL or an integer
chat	real value for overdispersion parameter
extractfn	function to extract output values from fitted model
data	optional list of simulated data saved from previous call to simulate.secr
tracelevel	integer for level of detail in reporting (0,1,2)
hessian	character string controlling the computation of the Hessian matrix
start	vector of starting 'beta' values for secr.fit
...	other arguments (not used)

Details

For each replicate, simulate.secr calls sim.popn to generate session- and group-specific realizations of the (possibly inhomogeneous) 2-D Poisson distribution fitted in object, across the habitat mask(s) in object. Group subpopulations are combined using rbind.popn within each session; information to reconstruct groups is retained in the individual-level factor covariate(s) of the resulting popn object (corresponding to object$groups). Each population is then sampled using the fitted detection model and detector (trap) array(s) in object.

The random number seed is managed as in simulate.lm.

simulate.secr does not yet work with models fitted using conditional likelihood (object$CL = TRUE). Detector type is determined by detector(traps(object$capthist)), which should be one of 'single', 'multi', 'proximity', 'areasearch' or 'count'.

sim.secr is a wrapper function. If data = NULL (the default) then it calls simulate.secr to generate new datasets. If data is provided then nsim is taken to be length(data). secr.fit is called to fit the original model to each new dataset. Results are summarized according to the user-provided function extractfn. The default extractfn returns the deviance and its degrees of freedom; a NULL value for extractfn returns the fitted secr objects after trimming to reduce bulk. Simulation uses the detector type of the data, even when another likelihood is fitted (this is the case with single-catch data, for which a multi-catch likelihood is fitted). Warning messages from secr.fit are suppressed.

extractfn should be a function that takes an secr object as its only argument.

tracelevel=0 suppresses most messages; tracelevel=1 gives a terse message at the start of each fit; tracelevel=2 also sets 'details\$trace = TRUE' for secr.fit, causing each likelihood evaluation to be reported.

It is OK (and faster) to use hessian='none' unless extractfn needs variances or covariances.

sim.capthist is a more direct way to simulate data from a null model (i.e. one with constant parameters for density and detection). It is limited to a single session.

Value

For simulate.secr, a list of data sets ('capthist' objects). This list has class=('list','secrdata'); the initial state of the random number generator (roughly, the value of .Random.seed) is stored as the attribute 'seed'.

The value from `sim.secr` depends on `extractfn`: if that returns a numeric vector of length `n.extract` then the value is a matrix with `dim = c(nsim, n.extract)` (i.e., the matrix has one row per replicate and one column for each extracted value). Otherwise, the value returned by `sim.secr` is a list with one component per replicate (strictly, an object of class = c('list','secrlist')). Each simulated fit may be retrieved *in toto* by specifying `extractfn = identity`, or slimmed down by specifying `extractfn = NULL` or `extractfn = trim`, which are equivalent.

For either form of output from `sim.secr` the initial state of the random number generator is stored as the attribute 'seed'.

Note

The value returned by `simulate.secr` is a list of 'capthist' objects; if there is more than one session, each 'capthist' is itself a sort of list .

The classes 'secrdata' and 'secrlist' are used only to override the ugly and usually unwanted printing of the seed attribute.

The default value for `start` in `sim.secr` is the previously fitted parameter vector. Alternatives are NULL or `object$start`.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`sim.capthist`, `secr.fit`, `simulate`

Examples

```
data(secrdemo)
simulate(secrdemo.0, nsim = 2)

## Not run:

## this would take a long time...
sims <- sim.secr(secrdemo.0, nsim = 99)
deviance(secrdemo.0)
devs <- c(deviance(secrdemo.0),sims$deviance)
quantile(devs, probs=c(0.95))
rank(devs)[1] / length(devs)

## to assess bias
extrfn <- function (object) unlist(predict(object)['D',-1])
sims <- sim.secr(secrdemo.0, nsim = 50, extractfn=extrfn)
sims

## with a larger sample, could get parametric bootstrap CI
quantile(sims[,1], c(0.025, 0.975))


## End(Not run)
```

`subset.capthist` *Subset or Split capthist Object*

Description

Create a new `capthist` object or list of objects by selecting rows (individuals), columns (occasions) and traps from an existing `capthist` object.

Usage

```
## S3 method for class 'capthist':
subset(x, subset = NULL, occasions = NULL, traps = NULL,
    sessions = NULL, cutval = NULL, dropnull = TRUE, dropunused =
    TRUE, renumber = FALSE, ...)
## S3 method for class 'capthist':
split(x, f, drop = FALSE, prefix = 'S', ...)
```

Arguments

`x`	object of class `capthist`
`subset`	vector of subscripts to select rows (individuals)
`occasions`	vector of subscripts to select columns (occasions)
`traps`	vector of subscripts to select detectors (traps)
`sessions`	vector of subscripts to select sessions
`cutval`	new threshold for signal strength
`dropnull`	logical for whether null (all-zero) capture histories should be dropped
`dropunused`	logical for whether never-used detectors should be dropped
`renumber`	logical for whether row.names should be replaced with sequence number in new `capthist`
`f`	factor or object that may be coerced to a factor
`drop`	logical indicating if levels that do not occur should be dropped (if f is a factor)
`prefix`	a character prefix to be used for component names when values of f are numeric
`...`	other arguments (not used currently)

Details

Subscript vectors may be either logical (length equal to the relevant dimension of `x`) or integer-valued. Subsetting is applied to attributes (e.g. `covariates`, `traps`) as appropriate. The default action is to include all rows, columns and traps if the relevant argument is omitted.

When `traps` is provided, detections at other detectors are set to zero, as if the detector had not been used, and the corresponding rows are removed from `traps`. If the detector type is 'proximity' then selecting traps also reduces the third dimension of the capthist array.

`split` generates a list in which each component is a `capthist` object. Each component corresponds to a level of `f`.

To combine (pool) occasions use `reduce.capthist`. There is no equivalent of `unlist` for lists of `capthist` objects.

Value

`capthist` object with the requested subset of observations, or a list of such objects (i.e., a multi-session `capthist` object). List input results in list output, except when a single session is selected.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`capthist`, `rbind.capthist`, `reduce.capthist`

Examples

```
tempcapt <- sim.capthist (make.grid(nx=6, ny=6), nocc=6)
summary(subset(tempcapt, occ=c(1,3,5)))

## Consider 'proximity' detections at a random subset of detectors
## This would not make sense for 'multi' detectors, as the
## excluded detectors influence detection probabilities in
## sim.capthist.

tempcapt2 <- sim.capthist (make.grid(nx = 6, ny = 6,
    detector = 'proximity'), nocc = 6)
tempcapt3 <- subset(tempcapt2, traps = sample(1:36, 18,
    replace=FALSE))
summary(tempcapt3)
plot(tempcapt3)

split (tempcapt2, f = sample (c('A','B'), nrow(tempcapt2),
    replace = TRUE))
```

`subset.mask` *Subset Mask Object*

Description

Retain selected rows of a `mask` object.

Usage

```
## S3 method for class 'mask':
subset(x, subset, ...)

## S3 method for class 'mask':
rbind(...)
```

Arguments

`x`	`mask` object
`subset`	numeric or logical vector to select rows of mask
`...`	two or more `mask` objects (rbind only)

Details

The subscripts in `subset` may be of type integer, character or logical as described in `Extract`.

Covariates are ignored by `rbind.mask`.

Value

For `subset`, an object of class 'mask' with only the requested subset of rows and 'type' attribute set to 'subset'.

For `rbind`, an object of class 'mask' with all unique rows from the masks in ..., and 'type' attribute set to 'rbind'.

Author(s)

Murray Efford `<murray.efford@otago.ac.nz>`

See Also

`mask`

Examples

```
tempmask <- make.mask(make.grid())
OK <- (tempmask$x + tempmask$y) > 100
tempmask <- subset(tempmask, subset = OK)
plot(tempmask)
```

---

subset.traps *Subset traps Object*

Description

Retain selected rows of a traps object.

Usage

```
## S3 method for class 'traps':
subset(x, subset, ...)
## S3 method for class 'traps':
split(x, f, drop = FALSE, prefix = 'S', ...)
```

Arguments

`x`	`traps` object
`subset`	vector to subscript the rows of `x`
`...`	arguments passed to other functions
`f`	factor or object that may be coerced to a factor
`drop`	logical indicating if levels that do not occur should be dropped (if f is a factor)
`prefix`	a character prefix to be used for component names when values of f are numeric

Details

The subscripts in `subset` may be of type integer, character or logical as described in `Extract`.

`split` generates a list in which each component is a `traps` object. Each component corresponds to a level of `f`. The argument 'x' of `split` cannot be a list.

Value

An object of class `traps` with only the requested subset of rows. Subsetting is applied to `usage` and `covariates` attributes if these are present.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`, `rbind.traps`

Examples

```
## odd-numbered traps only, using modulo operator
temptrap <- make.grid(nx = 7, ny = 7)
t2 <- subset(temptrap, as.logical(1:nrow(temptrap) %% 2))
plot(t2)
```

---

summary.capthist *Summarise Detections*

Description

Concise description of `capthist` object.

Usage

```
## S3 method for class 'capthist':
summary(object, ...)

## S3 method for class 'summary.capthist':
print(x, ...)

counts(CHlist, counts = 'M(t+1)')
```

Arguments

`object`	`capthist` object
`x`	`summary.capthist` object
`...`	arguments passed to other functions
`CHlist`	capthist object, especially a multi-session object
`counts`	character vector of count names

Details

These counts are reported by `summary.capthist`

n	number of individuals detected on each occasion
u	number of individuals detected for the first time on each occasion
f	number of individuals detected exactly f times
M(t+1)	cumulative number of individuals detected
losses	number of individuals reported as not released on each occasion
detections	number of detections, including within-occasion 'recaptures'
traps visited	number of detectors at which at least one detection was recorded
traps set	number of detectors, excluding any 'not set' in `usage` attribute of `traps` attribute

`counts` may be used to return the specified counts in a compact session x occasion table. If more than one count is named then a list is returned with one component for each type of count.

Value

An object of class `summary.capthist`, a list with at least these components

`detector`	detector type in {'single', 'multi', 'proximity'}
`ndetector`	number of detectors
`xrange`	range of x coordinates of detectors
`yrange`	range of y coordinates of detectors
`spacing`	mean distance from each trap to nearest other trap
`counts`	matrix of summary counts (rows) by occasion (columns). See Details.
`dbar`	mean recapture distance
`RPSV`	root pooled spatial variance

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`dbar`, `RPSV`, `capthist`

Examples

```
temptrap <- make.grid(nx = 5, ny = 3)
summary(sim.capthist(temptrap))
summary(sim.capthist(temptrap))$counts['n',]
```

summary.mask *Summarise Habitat Mask*

Description

Concise summary of a mask object.

Usage

```
## S3 method for class 'mask':
summary(object, ...)
## S3 method for class 'summary.mask':
print(x, ...)
```

Arguments

object	mask object
x	summary.mask object
...	other arguments (not used)

Details

The bounding box is the smallest rectangular area with edges parallel to the x- and y-axes that contains all points and their associated grid cells. A print method is provided for objects of class summary.mask.

Value

Object of class 'summary.mask', a list with components

detector	character string for detector type ('single','multi','proximity')
type	mask type ('traprect', 'trapbuffer', 'pdot', 'polygon', 'user', 'subset')
nmaskpoints	number of points in mask
xrange	range of x coordinates
yrange	range of y coordinates
meanSD	dataframe with mean and SD of x, y, and each covariate
spacing	nominal spacing of points
cellarea	area (ha) of grid cell associated with each point
bounding box	dataframe with x-y coordinates for vertices of bounding box
covar	summary of each covariate

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

mask

Examples

```
tempmask <- make.mask(make.grid())
## left to right gradient
covariates (tempmask) <- data.frame(x = tempmask$x)
summary(tempmask)
```

---

summary.traps *Summarise Detector Array*

Description

Concise description of `traps` object.

Usage

```
## S3 method for class 'traps':
summary(object, getspacing = TRUE, ...)
## S3 method for class 'summary.traps':
print(x, terse = FALSE, ...)
```

Arguments

`object`	`traps` object
`getspacing`	logical to calculate spacing of detectors from scratch
`x`	`summary.traps` object
`terse`	if TRUE suppress printing of usage and covariate summary
`...`	arguments passed to other functions

Details

When `object` includes both categorical (factor) covariates and `usage`, usage is tabulated for each level of the covariates.

Computation of `spacing` (mean distance to nearest trap) is slow and may hit a memory limit when there are many traps. In this case, turn off the computation with `getspacing` = FALSE.

Value

An object of class `summary.traps`, a list with elements

`detector`	detector type ('multi', 'proximity' etc.)
`ndetector`	number of detectors
`xrange`	range of x coordinates
`yrange`	range of y coordinates
`spacing`	mean distance from each trap to nearest other trap
`usage`	table of usage by occasion
`covar`	summary of covariates

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

print, traps

Examples

```
demo.traps <- make.grid()
summary(demo.traps)   ## uses print method for summary.traps object
```

---

traps *Detector Array*

Description

An object of class traps encapsulates a set of detector (trap) locations and related data. A method of the same name extracts or replaces the traps attribute of a capthist object.

Usage

```
traps(object, ...)
traps(object) <- value
```

Arguments

object	a capthist object.
value	traps object to replace previous.
...	other arguments (not used).

Details

An object of class traps holds detector (trap) locations as a data frame of x-y coordinates. Trap identifiers are used as row names. The required attribute 'detector' records the type of detector ('single', 'multi' or 'proximity').

Other possible attributes of a traps object are trap-specific covariates (covariates) and a matrix of binary (0/1) codes indicating whether each detector was used on each occasion (usage). If usage is specified, at least one detector must be 'used' on each occasion.

Note

Generic methods are provided to select rows (subset.traps), combine two or more arrays (rbind.traps), shift an array (shift.traps), and to rotate an array (rotate.traps). The attributes usage and covariates may be extracted or replaced using generic methods of the same name.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

References

Efford, M. G. (2007) *Density 4.1: software for spatially explicit capture–recapture*. Department of Zoology, 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 and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer, New York. Pp. 255–269.

See Also

make.grid, read.traps, plot.traps, secr.fit

Examples

```
demotraps <- make.grid(nx = 8, ny = 6, spacing = 30)
demotraps    ## uses print method for traps
summary (demotraps)

plot (demotraps, border = 50, label = TRUE, offset = 8,
    gridlines=FALSE)

## generate an arbitrary covariate 'randcov'
covariates (demotraps) <- data.frame(randcov = rnorm(48))

## overplot detectors that have high covariate values
temptr <- subset(demotraps, covariates(demotraps)$randcov > 0.5)
plot (temptr, add = TRUE,
    detpar = list (pch = 16, col = 'green', cex = 2))
```

---

traps.info *Detector Attributes*

Description

Extract or replace attributes of an object of class 'traps'.

Usage

```
polyID(object)
polyID(object) <- value
transectID(object)
transectID(object) <- value
searcharea(object)
searcharea(object) <- value
transectlength(object)
```

Arguments

object	a 'traps' object
value	replacement value (see Details)

Details

The 'polyID' and 'transectID' functions assign and extract the attribute of a 'traps' object that relates vertices (rows) to particular polygons or transects. The replacement value should be a factor of length equal to nrow(object).

The 'searcharea' attribute of a 'quadratbinary' or 'quadratcount' traps object is the area in hectares searched at each detector point (quadrat). Usually, this is the area of a rectangular pixel determined by the detector spacing (spacex, spacey). Replacement creates square pixels with dimensions spacex = spacey = value^0.5 * 100.

The 'searcharea' of a 'polygon' traps object is a vector of the areas of the component polygons. This is a read-only value (i.e. 'searcharea<-' does not apply).

The 'transectlength' of a 'transect' traps object is a vector of the lengths of the component transects in metres. This is a read-only value.

Value

`polyID` - a factor with one level per polygon. `searcharea` - numeric value of quadrat area or polygon areas, in hectares. `transectlength` - numeric value of transect lengths, in metres.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`

Examples

```
## default is a single polygon
temp <- make.grid(detector = 'polygon', hollow = TRUE)
polyID(temp)
plot(temp)

## split in two
temp <- make.grid(detector = 'polygon', hollow = TRUE)
polyID(temp) <- factor(rep(c(1,2),rep(10,2)))
plot(temp)
```

`trim` *Drop Unwanted List Components*

Description

Drop unwanted components from a `list` object, usually to save space.

Usage

```
## Default S3 method:
trim(object, drop, keep)
## S3 method for class 'secr':
trim(object, drop = c("mask", "design", "design0", "D"),
    keep = NULL)
```

Arguments

object	a list object
drop	vector identifying components to be dropped
keep	vector identifying components to be kept

Details

drop may be a character vector of names or a numeric vector of indices. If both drop and keep are given then the action is conservative, dropping only components in drop and not in keep.

Value

a list retaining selected components.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
data(secrdemo)
names(secrdemo.0)
names(trim(secrdemo.0))
object.size(secrdemo.0)
object.size(trim(secrdemo.0))
```

---

usage *Detector Usage*

Description

Extract or replace usage information of a traps object.

Usage

```
usage(object, ...)
usage(object) <- value
```

Arguments

`object`	a `traps` object
`value`	a matrix of traps x occasions 1 if trap[i] used on occasion[j], zero otherwise.
`...`	other arguments (not used)

Details

For replacement, the number of rows of `value` must match exactly the number of traps in `object`.

Value

usage(object) returns the usage matrix of the `traps` object. `usage(object)` may be NULL.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`traps`

Examples

```
demo.traps <- make.grid(nx = 6, ny = 8)
## random usage over 5 occasions
usage(demo.traps) <- matrix (sample(0:1, 48*5, replace = TRUE,
    p = c(0.5,0.5)), nc = 5)
usage(demo.traps)
summary(demo.traps)
```

`vcov.secr` *Variance - Covariance Matrix of SECR Parameters*

Description

Variance-covariance matrix of beta or real parameters from fitted secr model.

Usage

```
## S3 method for class 'secr':
vcov(object, realnames = NULL, newdata = NULL,
    byrow = FALSE, ...)
```

Arguments

`object`	secr object output from the function `secr.fit`
`realnames`	vector of character strings for names of 'real' parameters
`newdata`	dataframe of predictor values
`byrow`	logical for whether to compute covariances among 'real' parameters for each row of new data, or among rows for each real parameter
`...`	other arguments (not used)

Details

By default, returns the matrix of variances and covariances among the estimated model coefficients (beta parameters).

If `realnames` and `newdata` are specified, the result is either a matrix of variances and covariances for each 'real' parameter among the points in predictor-space given by the rows of `newdata` or among real parameters for each row of `newdata`. Failure to specify `newdata` results in a list of variances only.

Value

A matrix containing the variances and covariances among beta parameters on the respective link scales, or a list of among-parameter variance-covariance matrices, one for each row of `newdata`, or a list of among-row variance-covariance matrices, one for each 'real' parameter.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

vcov, secr.fit, print.secr

Examples

```
## Use previosuly fitted secr object
data(secrdemo)
vcov(secrdemo.0)
```

---

verify *Check SECR Data*

Description

Check that the data and attributes of an object are internally consistent to avoid crashing functions such as `secr.fit`

Usage

```
## Default S3 method:
verify(object, report, ...)
## S3 method for class 'traps':
verify(object, report = 2, ...)
## S3 method for class 'capthist':
verify(object, report = 2, tol = 0.01, ...)
## S3 method for class 'mask':
verify(object, report = 2, ...)
```

Arguments

`object`	an object of class 'traps', 'capthist' or 'mask'
`report`	integer code for level of reporting to the console. 0 = no report, 1 = errors only, 2 = full.
`tol`	numeric tolerance for deviations from transect line (m)
`...`	other arguments (not used)

Details

Checks are performed specific to the class of 'object'. The default method is called when no specific method is available (i.e. class not 'traps', capthist' or 'mask'), and does not perform any checks.

`verify.capthist`

1. No 'traps' component
2. Invalid 'traps' component reported by verify.traps
3. No live detections
4. Missing values not allowed in capthist
5. Live detection(s) after reported dead
6. More than one capture in single-catch trap(s)
7. More than one detection per detector per occasion at proximity detector(s)
8. Count detector counts less than zero
9. Signal detector signal(s) less than threshold or invalid threshold
10. Number of rows in 'traps' object not compatible with reported detections
11. Number of rows in dataframe of individual covariates differs from capthist
12. Number of occasions in usage matrix differs from capthist
13. Detections at unused detectors
14. Coordinates of detection(s) outside polygons ('polygon' detectors)
15. Coordinates of detection(s) do not lie on any transect ('transect' detectors)

`verify.traps`

1. Missing detector coordinates not allowed
2. Number of rows in dataframe of detector covariates differs from expected
3. Number of detectors in usage matrix differs from expected
4. Occasions with no used detectors
5. Area detectors, but quadrats overlap or no area specified

`verify.mask`

1. Valid x and y coordinates
2. Number of rows in covariates dataframe differs from expected

Earlier errors may mask later errors: fix & re-run.

Value

A list with the component `errors`, a logical value indicating whether any errors were found. If `object` contains multi-session data then session-specific results are contained in a further list component `bysession`.

Full reporting is the same as 'errors only' except that a message is posted when no errors are found.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

`capthist`, `secr.fit`

Examples

```
data(captdata)
verify(captdata)

## create null (complete) usage matrix, and mess it up
temptraps <- make.grid()
usage(temptraps) <- matrix(1, nr = nrow(temptraps), nc = 5)
usage(temptraps)[,5] <- 0
verify (temptraps)

## create mask, and mess it up
tempmask <- make.mask(temptraps)
verify(tempmask)
tempmask[1,1] <- NA
verify(tempmask)
```

---

write.capthist *Write Data to Text File*

Description

Export detections or detector layout to a text file in form suitable for DENSITY

Usage

```
write.capthist(object, file = "", ..., deblank = TRUE, header = TRUE,
    sess = '1', ndec = 2)

write.traps(object, file = "", ..., deblank = TRUE,  header = TRUE,
    ndec = 2)
```

Arguments

object	capthist or traps object
file	name of output file (character string)
...	other arguments passed to write.table
deblank	remove blanks from character string used to identify detectors?
header	output descriptive header?
sess	session identifier
ndec	number of digits after decimal point for x,y coordinates

Details

Existing file will be replaced without warning.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

Examples

```
data (captdata)
write.capthist (captdata)
```

Index