Package ‘ipsecr’

January 15, 2024

Type Package

Title Spatially Explicit Capture-Recapture by Inverse Prediction

Version 1.4.1

Date 2024-01-15

Description Estimates the density of a spatially distributed animal population sampled with an array of passive detectors, such as traps. Models incorporating distance-dependent detection are fitted by simulation and inverse prediction as proposed by Efford (2004) <doi:10.1111/j.0030-1299.2004.13043.x>.

Depends R (>= 3.5.0), secr (>= 4.5.8)

Imports graphics, grDevices, MASS, nlme, parallel, Rcpp (>= 1.0.8.3), stats, stringr, tools, utils

Suggests FrF2, knitr, plot3D , rmarkdown, sf, spatstat, testthat (>= 0.11.0)

LinkingTo BH, Rcpp, RcppArmadillo

VignetteBuilder knitr

License GPL (>= 2)

LazyData yes

LazyDataCompression xz

URL `https://github.com/MurrayEfford/ipsecr/`,

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

NeedsCompilation yes

Author Murray Efford [aut, cre] (<`https://orcid.org/0000-0001-5231-5184`>)

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

Repository CRAN

Date/Publication 2024-01-15 08:20:07 UTC

R topics documented:

`ipsecr-package` *Spatially Explicit Capture–Recapture by Inverse Prediction*

Description

Functions to estimate the density of a spatially distributed animal population sampled with an array of passive detectors, such as traps. **ipsecr** addresses 'difficult' models that strictly cannot be fitted by maximum likelihood in the package **secr** (Efford 2022). The classic example concerns discrete-time data from single-catch traps.

Details

Package:	ipsecr
Type:	Package
Version:	1.4.1
Date:	2024-01-15
License:	GNU General Public License Version 2 or later

Spatially explicit capture–recapture is a set of methods for studying marked animals distributed in space. Data comprise the locations of detectors (described in an object of class 'traps'), and the detection histories of individually marked animals. Individual histories are stored in an object of class 'capthist' that includes the relevant 'traps' object.

Models for population density (animals per hectare) and detection are defined in **ipsecr** using symbolic formula notation. The set of possible models overlaps with **secr** (some models for varying detection parameters are excluded, e.g., ~t, ~b). Density models may include spatial trend. Habitat is distinguished from nonhabitat with an object of class 'mask'.

Models are fitted in **ipsecr** by simulation and inverse prediction (Efford 2004, 2023). A model fitted with `ipsecr.fit` is an object of class `ipsecr`. Generic methods (plot, print, summary, etc.) are provided.

A link at the bottom of each help page takes you to the help index. The vignette includes worked examples.

The analyses in **ipsecr** extend those available in the software Density (see www.otago.ac.nz/density/ for the most recent version of Density). Help is available on the 'DENSITY | secr' forum at www.phidot.org and the Google group secrgroup. Feedback on the software is also welcome, including suggestions for additional documentation or new features consistent with the overall design.

'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 or lambda0 and sigma) by 'matching' statistics from `proxyfn(capthist)` (the target vector) to statistics from simulations of a 2-D population using the postulated detection model. Statistics (see Note) are defined by the proxy function, which should return a vector equal in length to the number of parameters (default np = 3). Simulations of the 2-D population use either internal C++ code or `sim.popn`. The simulated 2-D distribution of animals is Poisson by default.

The simulated population is sampled with internal C++ code, `sim.capthist`, or a user-specified function. Simulations match the detector type (e.g., 'single' or 'multi') and detector layout specified in traps(capthist), including allowance for varying effort if the layout has a `usage` attribute.

Simulations are usually 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.

A multivariate linear model is fitted to predict each vector of simulated proxies from the known parameter values. Simulations are performed at each design point until a specified precision is reached, up to a user-specified maximum number.

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.

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

If not provided, the starting values are determined automatically with the **secr** function `makeStart`.

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

If `ncores` > 1 the **parallel** package is used to create worker processes on multiple cores (see Parallel for more).

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. (2022) secr: Spatially explicit capture–recapture models. R package version 4.5.8. https://CRAN.R-project.org/package=secr/

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

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

`proxy.ms` `ipsecr.fit`, `secr.fit`, `capthist`, `mask`

`details` *Detail Specification for ipsecr.fit*

Description

The function `ipsecr.fit` allows many options. Some of these are used infrequently and have been bundled as a single argument `details` to simplify the documentation. They are described here in two groups: tuning parameters are listed in the following table, and more exotic options follow, listed alphabetically.

Tuning parameters

Parameter	Default	Description
boxsize1	0.2	scalar or vector of length np for size of design
boxsize2	0.05	as for `boxsize1`; used from second box onwards
centre	3	number of centre points in simulation design
min.nsim	20	minimum number of simulations per point
max.nsim	200	maximum number of simulations per point
dev.max	0.002	tolerance for precision of points in proxy space (see below)
var.nsim	2000	number of additional simulations to estimate variance-covariance matrix
min.nbox	2	minimum number of attempts to ‘frame’ solution
max.nbox	5	maximum number of attempts to ‘frame’ solution
max.ntries	2	maximum number of attempts at each simulation

`dev.max` defines a stopping rule: simulations are added in blocks of details$min.nsim until a measure of precision is less than dev.max for all proxies. If a vector of length 2, the first element applies to the first box and the second to all later boxes. The measure of precision may be the standard error on the link scale (details$boxtype = "absolute") or the coefficient of variation (details$boxtype = "relative").

Other 'details' components

`binomN` (default 0 = Poisson) integer code for distribution of counts.

`boxtype` (default "absolute") switches between specifying box size as additive on the transformed parameter scale ("absolute") and relative on the transformed parameter scale ("relative").

`CHmethod` (default "internal") chooses between internal C++ code, the **secr** function `sim.capthist`, and a user-provided R function with arguments "traps", "popn", "detectfn", "detectpar", and "noccasions". See also popmethod.

`contrasts` (default NULL) bmay be used to specify the coding of factor predictors. The value should be suitable for the 'contrasts.arg' argument of `model.matrix`. See 'Trend across sessions' in secr-multisession.pdf for an example.

`debug` (default FALSE) is used only for debugging. In ordinary use it should not be changed from the default.

`distribution` (default "poisson") specifies the distribution of the number of individuals detected n; 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 component 'distribution' may also take a numeric value larger than nrow(capthist), rather than "binomial" or "poisson".

`extraparam` is a list of starting values for extra 'real' parameters that may be needed for some user-specified models. See the vignette for explanation and an example.

`factorial` (default "full") chooses between "full" or "fractional" design. Fractional requires the package **FrF2** (Groemping 2014).

`forkonunix` (default TRUE) switches the cluster type generated by `makeCluster` between FORK and PSOCK.

`FrF2args` (default NULL) provides a list of arguments defining a fractional design.

`ignorenontarget` (default FALSE) may be used to ignore non-target information (the capthist attribute 'nontarget'). The default is to model non-target information if it is present.

`ignoreusage` (default FALSE) may be used to ignore usage (varying effort) information in the traps component. The default is to include usage in the model.

`keep.sim` (default FALSE) when TRUE causes `ipsecr.fit` to save additional output, specifically lists (one component per box) of the simulations and parameters for each box, and the final variance simulations.

`newdetector` (default NULL) may be used to override the detector type of the traps object embedded in the capthist passed to `ipsecr.fit`.

`Nmax` (default 1e4) maximum allowed population size for simulations.

`nontargettype` (default "exclusive") chooses among options "exclusive", "truncated", "erased, "independent", and "dependent". See vignette.

`popmethod` (default "internal") chooses between internal C++ code, the **secr** function `sim.popn`, and a user-provided R function with arguments 'mask', 'D' (density per cell of mask) and 'N' (number of individuals to simulate). See also CHmethod.

`savecall` (default TRUE) determines whether the function call is included in the output.

`YonX` (default TRUE) switches between regression of simulated proxy values Y on controlled parameter values X, or the reverse (which is not fully implemented).

References

Groemping, U. (2014). R Package FrF2 for Creating and Analyzing Fractional Factorial 2-Level Designs. *Journal of Statistical Software*, **56**, 1–56. https://www.jstatsoft.org/article/view/v056i01.

See Also

`ipsecr.fit`

`Internal` *Internal Functions*

Description

Functions called internally by **ipsecr**. These are exported and may be called separately for testing.

Usage

```
proxy.ms(capthist, model = NULL, trapdesigndata = NULL, ...)

detectionDesignData(capthist, byoccasion = FALSE, ...)

proxyfn1(capthist, N.estimator =  c("n", "null","zippin","jackknife"), ...)

simpop(mask, D, N, details = list(), ...)

simCH(traps, popn, detectfn, detparmat, noccasions, NT = NULL, details =
    list(), ...)

rpsv(capthist)
rpsvi(capthist)
```

Arguments

`capthist`	secr capthist object
`model`	named list of model formulae (see `ipsecr.fit`)
`trapdesigndata`	dataframe with one row for each detector and session

`...`	other arguments, mostly unused
`byoccasion`	logical; if TRUE the output rows are repeated for each occasion
`N.estimator`	character name of closed-population estimator
`mask`	secr mask object
`D`	numeric density in each mask cell
`N`	integer number of animals to simulate
`traps`	detector locations as **secr** traps object
`popn`	animal locations as **secr** popn object
`detectfn`	integer code for detection function (see detectfn)
`detparmat`	numeric matrix of detection parameter values
`noccasions`	integer number of sampling occasions
`NT`	numeric hazard of non-target interference at each detector
`details`	list with optional additional named arguments

Details

`proxy.ms` is the default proxyfn used by `ipsecr.fit`. When used internally by `ipsecr.fit`, 'model' and 'trapdesigndata' are passed automatically. The `...` argument of `proxy.ms` may be used to pass arguments to `addCovariates`, especially 'spatialdata'. Function `detectionDesignData` is used internally to construct design data for non-constant detection models (lambda0, sigma), used in the glm 'data' argument. The capthist argument for `detectionDesignData` should always be a list (wrap a single-session capthist in list()).

`simpop` is used by `ipsecr.fit` for popmethod 'internal'. It is faster and simpler than the **secr** function `sim.popn`. The details component 'distribution' is a character value that may be 'poisson' (default) or 'even.

`simCH` is used by `ipsecr.fit` for CHmethod 'internal'. It is faster and simpler than the **secr** function `sim.capthist`, and optionally simulates non-target interference. The argument `detparmat` is an individual x parameter matrix, with parameters in the order usual for `detectfn`.

`D` and `NT` are matrices with one column per session.

`proxyfn1` is a simple proxy function included mostly for historical reasons. It updates the function of Efford (2004) by log-transforming N, using a complementary log-log transformation instead of odds for p, and using log(RPSV(capthist)) for sigma. If you're interested, look at the code.

`rpsv(capthist)` is equivalent to **secr** RPSV(capthist, CC = TRUE). `rpsvi(capthist)` returns a vector of individual-specific rpsv.

Value

proxy.ms – a numeric vector of length >= 3 corresponding to proxies for a wide range of models, including multi-session density and non-target interference models.

detectionDesignData – a dataframe with one row per individual per session (byoccasion = FALSE) or one row per individual per occasion per session (byoccasion = TRUE), ordered by session, occasion and individual. Columns include x and y coordinates of the individual's centroid, session, and any individual covariates.

proxyfn1 – a numeric vector of length 3 corresponding to proxies for population size, capture probability intercept and scale of detection.

simpop – a `popn` object.

simCH – a single-session `capthist` object.

rpsv – scalar

rpsvi – vector, one element per animal

Note

`proxyfn0` was removed in version 1.2.0.

References

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

See Also

`ipsecr.fit`, `plotProxy`

Examples

```
proxy.ms(captdata)
```

`ipsecr-defunct` *Defunct Functions in Package* **ipsecr**

Description

These functions are no longer available in **ipsecr**.

Usage

```
# Defunct in 1.2.0 (2022-08)

proxyfn0()
```

Details

proxyfn0 was removed without warning in ipsecr 1.2.0. Use `proxyfn1` or `proxy.ms`.

See Also

`ipsecr-deprecated`

`ipsecr-deprecated` *Deprecated Functions in Package* **ipsecr**

Description

These functions will be removed from future versions of **ipsecr**.

Details

No functions are deprecated at this point.

See Also

`ipsecr-defunct`,

`ipsecr.fit` *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.

Usage

```
ipsecr.fit(capthist, proxyfn = proxy.ms, model = list(D ~ 1, g0 ~ 1, sigma ~ 1),
    mask = NULL, buffer = 100, detectfn = "HN", binomN = NULL, start = NULL,
    link = list(), fixed = list(), timecov = NULL, sessioncov = NULL,
    details = list(), verify = TRUE, verbose = TRUE, ncores = NULL,
    seed = NULL, ...)
```

Arguments

`capthist`	secr capthist object including capture data and detector (trap) layout
`proxyfn`	function to compute proxy from capthist for each coefficient (beta parameter)
`model`	list with optional components each symbolically defining a linear predictor for one real parameter using `formula` notation
`mask`	`mask` object
`buffer`	scalar mask buffer radius in metres if `mask` not specified
`detectfn`	integer code or character string for shape of detection function 0 = halfnormal, 1 = hazard rate etc. – see detectfn
`binomN`	integer code for distribution of counts (see Details)

`start`	vector of initial values for beta parameters, or `ipsecr` object from which they may be derived
`link`	list with optional components corresponding to 'real' parameters (e.g., 'D', 'g0', 'sigma'), each a character string in {"log", "logit", "identity", "sin"} for the link function of one real parameter
`fixed`	list with optional components corresponding to real parameters giving the scalar value to which the parameter is to be fixed
`timecov`	optional dataframe of values of time (occasion-specific) covariate(s). NOT USED
`sessioncov`	optional dataframe of values of session-specific covariate(s)
`details`	list of additional settings, to control estimation (see Details)
`verify`	logical, if TRUE the input data are checked with `verify`
`verbose`	logical, if TRUE then messages are output during execution
`ncores`	integer number of cores to use for parallel processing
`seed`	either NULL or an integer that will be used in a call to `set.seed`
`...`	other arguments passed to proxy function

Details

The vignette should be consulted for a full exposition.

Parallel computation: `ncores` determines the number of worker processes in a cluster created by `makeCluster` (default type "FORK" on Unix platforms, otherwise "PSOCK"). If `ncores = NULL` this defaults to the value from `setNumThreads`. Simulations are distributed over worker processes using `parRapply`. There are substantial overheads in running multiple processes: using too many will slow down fitting. With PSOCK clusters (i.e. on Windows) fitting is very often fastest with ncores = 1.

The 'details' argument: `details` is used for various specialized settings listed below. These are also described separately - see `details`.

Name	Default	Description
boxsize1	0.2	scalar or vector of length np for size of design
boxsize2	0.05	as for `boxsize1`; used from second box onwards
boxtype	'absolute'	'absolute' or 'relative'
centre	3	number of centre points in simulation design
dev.max	0.002	tolerance for precision of points in predictor space
var.nsim	2000	number of additional simulations to estimate variance-covariance matrix
keep.sim	FALSE	if true then the variance simulations are saved
min.nsim	20	minimum number of simulations per point
max.nsim	200	maximum number of simulations per point
min.nbox	2	minimum number of attempts to 'frame' solution
max.nbox	5	maximum number of attempts to 'frame' solution
max.ntries	2	maximum number of attempts at each simulation
distribution	'poisson'	'poisson', 'binomial' or 'even'
binomN	0	integer code for distribution of counts (unused)
ignorenontarget	FALSE	override nontarget attribute of capthist

ignoreusage	FALSE	override usage in traps object of capthist
debug	FALSE	stop at arbitrary points in execution (varies)
savecall	TRUE	optionally suppress saving of call
newdetector	NULL	detector type that overrides detector(traps(capthist))
contrasts	NULL	coding of factor predictors
popmethod	‘internal’	‘internal’ or ‘sim.popn’ or a user-provided function
CHmethod	‘internal’	‘internal’ or ‘sim.capthist’ or a user-provided function
factorial	‘full’	‘full’ or ‘fractional’ design
FrF2args	NULL	arguments for FrF2 when factorial = ’fractional’
extraparam	NULL	list of starting values for extra parameters (see vignette)
forkonunix	TRUE	logical choice between FORK and PSOCK cluster types (not Windows)

Value

An object of class ’ipsecr’, a list comprising:

`call`	the function call (if details$savecall)
`capthist`	input
`proxyfn`	input
`model`	input
`mask`	input
`detectfn`	input
`start`	input
`link`	input
`fixed`	input
`timecov`	input
`sessioncov`	input
`details`	input
`designD`	list of design data for density
`trapdesigndata`	list of design data for trap-specific models
`parindx`	mapping of coefficients (beta parameters) to real parameters
`vars`	names of covariates in model
`betanames`	names of coefficients
`realnames`	names of ’real’ parameters
`code`	integer completion code: 1 successful, 2 target not within final box, 3 exceeded maximum simulations
`beta`	estimates of coefficients on link scale
`beta.vcov`	variance-covariance matrix of estimates
`designbeta`	vertices of final box (design points)
`sim.lm`	last lm model fit
`ip.nsim`	total number of simulations

`var.nsim.OK`	number of successful variance simulations
`simulations`	optional simulation output (see details$keep.sim)
`parameters`	optional simulation input (see details$keep.sim)
`variance.bootstrap`	dataframe summarising simulations for variance estimation
`version`	package version
`starttime`	time execution started
`proctime`	processor time (seconds)
`seed`	RNG state

(The order and composition of the output list may change).

References

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

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

`proxy.ms`, `predict.ipsecr`, `summary.ipsecr`

Examples

```
ipsecrdemo <- ipsecr.fit(captdata, ncores = 1, buffer = 100, detectfn = 14, seed = 1237)
```

`ipsecrdemo` *SECR Model Fitted to Demonstration Data*

Description

Demonstration data from program Density are provided as a `capthist` object (`captdata`) ready for input to `ipsecr.fit`.

The fitted models are objects of class `ipsecr` formed by

`ipsecrdemo <- ipsecr.fit(captdata, ncores = 1, detectfn = 'HHN', seed = 1237, details = list(keep.sim = TRUE))`

Usage

```
data(ipsecrdemo)
```

Details

The raw data are 235 fictional captures of 76 animals over 5 occasions in 100 single-catch traps 30 metres apart on a square grid with origin at (365,365).

The fitted model uses a hazard halfnormal detection function and default values of other arguments.

Object	Description
ipsecrdemo	fitted ipsecr model – null

References

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

See Also

`capthist`, `read.capthist`, `secrdemo`

Examples

```
predict(ipsecrdemo)
```

`makeNewData` *Create Default Design Data*

Description

Internal function used to generate a dataframe containing design data for the base levels of all predictors in an secr object.

Usage

```
## S3 method for class 'ipsecr'
makeNewData(object, all.levels = FALSE, ...)
```

Arguments

`object`	fitted ipsecr model object
`all.levels`	logical; if TRUE then all levels of factors are included
`...`	other arguments (not used)

Details

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

Value

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

See Also

`predict.ipsecr`, `ipsecr.fit`

Examples

```
## from previously fitted model
makeNewData(ipsecrdemo)
```

`plot.ipsecr` *Plot Detection Functions*

Description

Plot detection functions using estimates of parameters in an ipsecr object.

Usage

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

Arguments

`x`	an `ipsecr` 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`

Details

`newdata` is usually NULL, in which case one curve is plotted for each session and group. Otherwise, `predict.ipsecr` 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'.

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 of g(r) is assumed to be normal on the logit scale for non-hazard functions (detectfn 0:13). For hazard detection functions (detectfn 14:18) the hazard is assumed (from version 3.1.1) to be distributed normally on the log scale. Limits are back-transformed to the probability scale g(r).

Value

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

See Also

`Detection functions`, `plot`, `ipsecr`, `detectfnplot`

Examples

```
plot (ipsecrdemo, xval = 0:100, ylim = c(0, 0.4))
```

`plot3D.IP` *Plot design and saved simulations for one box from a model fitted with* `ipsecr.fit`

Description

A 3-D depiction of the design (a box in parameter space) and the resulting simulations (in proxy space).

Usage

```
plot3D.IP(object, box = 1, oldplot = NULL, plotcentre = TRUE, plotfinal = FALSE,
    zkludge = -0.2)
```

Arguments

`object`	ipsecr object from `ipsecr.fit` with details$keep.sim = TRUE
`box`	integer number of box to plot
`oldplot`	list containing transofrmations and plot limits from a previous execution
`plotcentre`	logical; if TRUE the centrepoint of the design box is plotted
`plotfinal`	logical; if TRUE the final estimates are plotted as a point in parameter space
`zkludge`	numeric adjustment for base value of z when plotfinal is TRUE

Details

The function is restricted to single-session models with 3 real parameters.

A 2-panel plot is generated, so the graphics options should allow at least 2 panels (e.g., `par(mfrow = c(1,2))`.

Parameters are plotted on the link scale.

The package **plot3D** is used (Soetaert 2021).

Value

Invisibly returns a list comprising

`pmatparm`	pmat used by **plot3D** for parameter space
`pmatsim`	pmat used by **plot3D** for proxy space
`pr`	2-row matrix with lower and upper plot limits of each parameter
`sr`	2-row matrix with lower and upper plot limits of each simulated proxy

References

Soetaert, K. (2021). plot3D: Plotting Multi-Dimensional Data. R package version 1.4. https://CRAN.R-project.org/package=plot3D

See Also

`ipsecr.fit`

Examples

```
if (requireNamespace("plot3D")) {
    par(mfrow = c(2,2), oma = c(1,1,3,1))
    # plot first box, saving projection and limits for later use
    oldplot <- plot3D.IP(ipsecrdemo, box = 1)
    # plot second box, using projections and limits from first box
    plot3D.IP(ipsecrdemo, box = 2, oldplot, plotfinal = TRUE, zkludge = -0.1)
    mtext(outer = TRUE, side = 3, line = 0.5, adj = c(0.2,0.8), cex = 1.1,
        c('Parameter space', 'Proxy space'))
}
```

`plotProxy` *Simulate and plot the relationship between a parameter and its designated proxy*

Description

As described in the vignette, each parameter is matched to a proxy value computed cheaply from the rawdata by the proxy function. This function provides a visual check on that relationship.

Usage

```
plotProxy(parameter = "sigma", proxyfn = proxy.ms, traps, mask, detectfn = "HHN",
    noccasions = 5, basepar = list(), xvals = NULL, nrepl = 100, add = FALSE,
    trend = TRUE, points = FALSE, boxplot = TRUE, boxplotargs = list(),
    link = "log", details = NULL, ...)
```

Arguments

`parameter`	character parameter of interest
`proxyfn`	function to compute vector of proxy values from a capthist object
`traps`	traps object
`mask`	habitat mask object
`detectfn`	numeric or character code for detection function (see detectfn)
`noccasions`	integer number of sampling occasions
`basepar`	named list with central values of parameters
`xvals`	specified values of focal paramater to simulate (optional)
`nrepl`	integer number of simulations
`add`	logical; if TRUE any plot is added to an existing plot
`trend`	logical; if TRUE a least-squares trend line is plotted
`points`	logical; if TRUE each simulated value is plotted
`boxplot`	logical; if TRUE a boxplots is plotted for each level of the focal parameter
`boxplotargs`	list of arguments for `boxplot` (optional)
`link`	character link function for transforming parameter x-axis
`details`	not used
`...`	other arguments passed to plot()

Details

Simulation uses the internal functions `simpop` and `simCH`.

`boxplotargs` may be used to override or add to the arguments of `boxplot`.

This version of `plotProxy()` does not allow for interference (NT) and assumes a simple SECR model with only 3 or 4 coefficients corresponding to density D and the parameters of the detection model (lambda0 or g0, sigma and possibly z).

Matching of proxies at the level of 'beta' parameters may be enabled in a future version.

Value

The simulated proxy values are returned invisibly as a matrix (nrepl x nlevels).

See Also

`proxy.ms`

Examples

```
# try with small number of replicates
trps <- traps(captdata)
msk <- make.mask(trps, buffer = 100)
base <- list(D = 5, lambda0 = 0.2, sigma = 25)
out <- plotProxy (parameter = 'D', traps = trps, mask = msk,
     basepar = base, boxplotargs = list(col='orange'), nrepl = 20)
```

`predict.ipsecr` *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 'ipsecr'
predict(object, newdata = NULL, type = c("response", "link"),
    se.fit = TRUE, alpha = 0.05, savenew = FALSE, ...)
```

Arguments

`object`	`ipsecr` object output from `ipsecr.fit`
`newdata`	optional dataframe of values at which to evaluate model
`type`	character; type of prediction required. The default ("response") provides estimates of the 'real' parameters.
`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
`...`	other arguments passed to `newdata`

Details

The variables in the various linear predictors are described in secr-models.pdf 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. The argument 'all.levels' may be passed to `newdata`; if TRUE then the default newdata includes all factor levels.

`realnames` may be used to select a subset of parameters.

Standard errors for parameters on the response (real) scale are by the delta method (Lebreton et al. 1992), and confidence intervals are backtransformed from the link scale.

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.

For `detectpar`, a list with the estimated values of detection parameters (e.g., g0 and sigma if detectfn = "halfnormal"). In the case of multi-session data the result is a list of lists (one list per session).

Note

`predictDsurface` should be used for predicting density at many points from a model with spatial variation. This deals automatically with scaling of x- and y-coordinates, and is much faster than predict.ipsecr. The resulting Dsurface object has its own plot method.

The argument 'scaled' was removed from both predict methods in version 2.10 as the scaleg0 and scalesigma features had been superceded by other parameterisations.

References

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

See Also

`ipsecr.fit`, `predictDsurface`

Examples

```
predict (ipsecrdemo)
```

`print.ipsecr` *Print or Summarise ipsecr Object*

Description

Print results from fitting a spatially explicit capture–recapture model or generate a list of summary values.

Usage

```
## S3 method for class 'ipsecr'
 print(x, newdata = NULL, alpha = 0.05, call = TRUE, ...)
## S3 method for class 'ipsecr'
 summary(object, newdata = NULL, alpha = 0.05, ...)
```

Arguments

`x`	`ipsecr` object output from `ipsecr.fit`
`object`	`ipsecr` object output from `ipsecr.fit`
`newdata`	optional dataframe of values at which to evaluate model
`alpha`	alpha level
`call`	logical; if TRUE the call is printed
`...`	other arguments (not used)

Details

Results from `print.ipsecr` 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 (optional)
version,time	ipsecr version, date and time fitting started, and elapsed time
Detector type	‘single’, ‘multi’, ‘proximity’ etc.
Detector number	number of detectors
Average spacing	
x-range	
y-range	
New detector type	as fitted when details$newdetector specified
N animals	number of distinct animals detected
N detections	number of detections
N occasions	number of sampling occasions
Mask area	
Model	model formula for each ‘real’ parameter
Fixed (real)	fixed real parameters
Detection fn	detection function type (halfnormal or hazard-rate)
Distribution	spatial model (details$distribution)
N parameters	number of parameters estimated
Design points	number of vertices and centre points
Simulations per box	total number
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

Value

The `summary` method constructs a list of outputs similar to those printed by the `print` method, but somewhat more concise and re-usable:

versiontime	ipsecr version, and date and time fitting started
traps	detector summary
capthist	capthist summary
mask	mask summary
modeldetails	miscellaneous model characteristics
coef	table of fitted coefficients with CI
predicted	predicted values (‘real’ parameter estimates)

See Also

`ipsecr.fit`

Examples

```
## load & print previously fitted null (constant parameter) model

print(ipsecrdemo)

summary(ipsecrdemo)
```

`vcov.ipsecr` *Variance - Covariance Matrix of SECR Parameters*

Description

Variance-covariance matrix of beta or real parameters from fitted ipsecr model.

Usage

```
## S3 method for class 'ipsecr'
vcov(object, realnames = NULL, newdata = NULL,
    byrow = FALSE, ...)
```

Arguments

`object`	ipsecr object output from the function `ipsecr.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.

See Also

`vcov`, `ipsecr.fit`, `print.ipsecr`

Examples

```
## previously fitted ipsecr model
vcov(ipsecrdemo)
```

Index