Package ‘openCR’

April 7, 2018

Type Package

Title Open Population Capture-Recapture

Version 1.1.0

Depends R (>= 3.2.0), secr (>= 3.1.5)

Imports utils, MASS, nlme, parallel, stats, Rcpp (>= 0.12.14), stringr, plyr, abind, methods

LinkingTo Rcpp

Suggests knitr, RMark, rgdal

VignetteBuilder knitr

Date 2018-04-07

Description Functions for non-spatial and spatial open-population capture-recapture analysis.

License GPL (>=2)

LazyData yes

LazyDataCompression xz

R topics documented:

`openCR-package` *Open Population Capture–Recapture Models*

Description

Functions for non-spatial open population analysis by Cormack-Jolly-Seber (CJS) and Jolly-Seber-Schwarz-Arnason (JSSA) methods, and by spatially explicit extensions of these methods. The parameterisation of JSSA recruitment is flexible (options include population growth rate λ, per capita recruitment f and seniority γ). Spatially explicit analyses may assume home-range centres are fixed or allow dispersal between primary sessions according to a normal, exponential or user-defined kernel.

Details

Package:	openCR
Type:	Package
Version:	1.1.0
Date:	2018-04-07
License:	GNU General Public License Version 2 or later

Data are observations of marked individuals from a 'robust' sampling design (Pollock 1982). Primary sessions may include one or more secondary sessions. Detection histories are assumed to be stored in an object of class 'capthist' from the package **secr**. Grouping of occasions into primary and secondary sessions is coded by the 'intervals' attribute (zero for successive secondary sessions).

A few test datasets are provided (`microtusCH`, `FebpossumCH`, `dipperCH`, `gonodontisCH`, `fieldvoleCH`) and some from **secr** are also suitable e.g. `ovenCH`.

Models are defined using symbolic formula notation. Possible predictors for include both predefined variables (b, session etc.) corresponding to 'behaviour' and other effects), and user-provided covariates.

Models are fitted by numerically maximizing the likelihood. The function `openCR.fit` creates an object of class `openCR`. Generic methods (print, AIC, etc.) are provided for each object class.

A link at the bottom of each help page takes you to the help index. The help pages are also available as `../doc/openCR-manual.pdf`.

See openCR-vignette.pdf for more.

Author(s)

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

References

Pledger, S., Pollock, K. H. and Norris, J. L. (2010) Open capture–recapture models with heterogeneity: II. Jolly-Seber model. *Biometrics* **66**, 883–890.

Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* **52**, 703–709.

Schwarz, C. J. and Arnason, A. N. (1996) A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics* **52**, 860–873.

See Also

`openCR.fit`, `capthist`, `ovenCH`

Examples

```
## a CJS model is fitted by default
openCR.fit(ovenCH)
```

`age.matrix` *Session-specific Ages*

Description

A matrix showing the age of each animal at each secondary session (occasion).

Usage

```
age.matrix(capthist, initialage = 0, minimumage = 0, maximumage = 1, collapse = FALSE)
```

Arguments

`capthist`	single-session capthist object
`initialage`	numeric or character name of covariate with age at first detection (optional)
`minimumage`	integer minimum age
`maximumage`	integer maximum age
`collapse`	logical; if TRUE then values for each individual are collapsed as a string with no spaces

Details

`age.matrix` is used by `openCR.design` for the predictors 'age' and 'Age'.

Computations use the intervals attribute of `capthist`, which may be non-integer.

Ages are inferred for occasions before first detection, back to the minimum age.

Value

Either a numeric matrix with dimensions (number of animals, number of secondary occasions) or if `collapse = TRUE` a character matrix with one column.

See Also

openCR.design

Examples

```
age.matrix(join(ovenCH), maximumage = 2, collapse = TRUE)
```

AIC.openCR *Compare openCR 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 'openCR'
AIC(object, ..., sort = TRUE, k = 2, dmax = 10,  use.rank = FALSE,
                         svtol = 1e-5, criterion = c('AIC','AICc'))

## S3 method for class 'openCRlist'
AIC(object, ..., sort = TRUE, k = 2, dmax = 10,  use.rank = FALSE,
                         svtol = 1e-5, criterion = c('AIC','AICc'))

## S3 method for class 'openCR'
logLik(object, ...)
```

Arguments

object	openCR object output from the function openCR.fit, or openCRlist
...	other openCR 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
use.rank	logical; if TRUE the number of parameters is based on the rank of the Hessian matrix
svtol	minimum singular value (eigenvalue) of Hessian used when counting non-redundant parameters
criterion	character, criterion to use for model comparison and weights

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 dAIC > `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 AIC.

`model`	character string describing the fitted model
`npar`	number of parameters estimated
`rank`	rank of Hessian
`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 AIC
`AICwt`	AICc model weight

`logLik.openCR` returns an object of class 'logLik' that has attribute `df` (degrees of freedom = number of estimated parameters).

Note

The default criterion is AIC, not AICc as in **secr** 3.1.

Computed values differ from MARK for various reasons. MARK uses the number of observations, not the number of capture histories when computing AICc. It is also likely that MARK will count parameters differently.

It is not be meaningful to compare models by AIC if they relate to different data.

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

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

`AIC`, `openCR.fit`, `print.openCR`, `LR.test`

Examples

```
## Not run:
m1 <- openCR.fit(ovenCH, type = 'JSSAf')
m2 <- openCR.fit(ovenCH, type = 'JSSAf', model = list(p~session))
AIC(m1, m2)

## End(Not run)
```

`cloned.fit` *Cloning to Evaluate Identifiability*

Description

The identifiability of parameters may be examined by refitting a model with cloned data (each capture history replicated `nclone` times). For identifiable parameters the estimated variances are proportional to `1/nclone`.

Usage

```
cloned.fit(object, nclone = 100, newdata = NULL, linkscale = FALSE)
```

Arguments

`object`	previously fitted openCR object
`nclone`	integer number of times to replicate each capture history
`newdata`	optional dataframe of values at which to evaluate model
`linkscale`	logical; if TRUE then comparison uses SE of linear predictors

Details

The key output is the ratio of SE for estimates from the uncloned and cloned datasets, adjusted for the level of cloning (`nclone`). For identifiable parameters the ratio is expected to be 1.0.

Cloning is not implemented for spatial models.

The comparison may be done either on the untransformed scale (using approximate SE) or on the link scale.

Value

Dataframe with columns* –

`estimate`	original estimate
`SE.estimate`	original SE
`estimate.xxx`	cloned estimate (xxx = nclone)
`SE.estimate.xxx`	cloned SE
`SE.ratio`	SE.estimate / SE.estimate.xxx / sqrt(nclone)

* 'estimate' becomes 'beta' when `linkscale = TRUE`.

References

Lele, S.R., Nadeem, K. and Schmuland, B. (2010) Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association* **105**, 1617–1625.

See Also

`openCR.fit`

Examples

```
fit <- openCR.fit(dipperCH)
cloned.fit(fit)
```

---

derived — *Derived Parameters From openCR Models*

Description

For ..CL openCR models, compute the superpopulation size or density. For all openCR models, compute the time-specific population size or density from the estimated superpopulation size and the turnover parameters.

Usage

```
## S3 method for class 'openCR'
derived(object, Dscale = 1, HTbysession = FALSE, ...)
## S3 method for class 'openCRlist'
derived(object, Dscale = 1, HTbysession = FALSE, ...)
openCR.esa(object, bysession = FALSE)
openCR.pdot(object, bysession = FALSE)
```

Arguments

`object`	fitted openCR model
`Dscale`	numeric to scale density
`HTbysession`	logical; Horvitz-Thompson estimates by session (see Details)
`...`	other arguments (not used)
`bysession`	logical; if TRUE then esa or pdot is computed separately for each session

Details

Derived estimates of density and superD are multiplied by `Dscale`. Use `Dscale = 1e4` for animals per 100 sq. km. `openCR.esa` and `openCR.pdot` are used internally by `derived.openCR`.

If `HTbysession` then a separate H-T estimate is derived for each primary session; otherwise a H-T estimate of the superpopulation is used in combination with turnover parameters (phi, beta) to obtain session-specific estimates. Results are often identical.

The output is an object with its own print method (see `print.derivedopenCR`).

The code does not yet allow user-specified newdata.

Value

`derived` returns an object of class c("derivedopenCR","list"), list with components

`totalobserved`	number of different individuals detected
`parameters`	character vector; names of parameters in model (excludes derived parameters)
`superN`	superpopulation size (non-spatial models only)
`superD`	superpopulation density (spatial models only)
`estimates`	data frame of counts and estimates
`Dscale`	numeric multiplier for printing densities

`openCR.pdot` returns a vector of experiment-wide detection probabilities under the fitted model (one for each detected animal).

`openCR.esa` returns a vector of effective sampling areas under the fitted model (one for each detected animal).

See Also

`openCR.fit`, `print.derivedopenCR`

Examples

```
# override default method to get true ML for L1
L1CL <- openCR.fit(ovenCH, type = 'JSSAlCL', method = 'Nelder-Mead')
predict(L1CL)
derived(L1CL)

## Not run:
## compare to above
L1 <- openCR.fit(ovenCH, type = 'JSSAl', method = 'Nelder-Mead')
predict(L1)
derived(L1)
```

```
## End(Not run)
```

---

dipperCH *Dippers*

Description

Lebreton et al. (1992) demonstrated Cormack-Jolly-Seber methods with a dataset on European Dipper (*Cinclus cinclus*) collected by Marzolin (1988) and the data have been much used since then. Dippers were captured annually over 1981–1987. We use the version included in the RMark package (Laake 2013).

Usage

```
dipperCH
```

Format

The format is a single-session secr capthist object. As these are non-spatial data, the traps attribute is NULL.

Details

Dippers were sampled in 1981–1987.

Source

MARK example dataset ‘ed.inp’. Also RMark (Laake 2013). See Examples.

References

Laake, J. L. (2013). *RMark: An R Interface for Analysis of Capture–Recapture Data with MARK*. AFSC Processed Report 2013-01, 25p. Alaska Fisheries Science Center, NOAA, National Marine Fisheries Service, 7600 Sand Point Way NE, Seattle WA 98115.

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.

Marzolin, G. (1988) Polygynie du Cincle plongeur (*Cinclus cinclus*) dans les c?tes de Lorraine. *L’Oiseau et la Revue Francaise d’Ornithologie* **58**, 277–286.

See Also

read.inp

Examples

```
m.array(dipperCH)

## Not run:

# From file 'ed.inp' in MARK input format
datadir <- system.file('extdata', package = 'openCR')
dipperCH <- read.inp(paste0(datadir, '/ed.inp'), grouplabel='sex',
    grouplevels = c('Male','Female'))
sessionlabels(dipperCH) <- 1981:1987   # labels only

# or extracted from the RMark package with this code
library(RMark)
MarkPath <- 'c:/MARK'              # customize as necessary
data(dipper)                       # retrieve dataframe of dipper capture histories
dipperCH2 <- unRMarkInput(dipper)  # convert to secr capthist object
sessionlabels(dipperCH2) <- 1981:1987   # labels only

# The objects dipperCH and dipperCH2 differ in the order of factor levels for 'sex'

## End(Not run)
```

`Field vole` *Kielder Field Voles*

Description

Captures of *Microtus agrestis* on a large grid in a clearcut within Kielder Forest, northern England, June–August 2000 (Ergon and Gardner 2014). Robust-design data from four primary sessions of 3–5 secondary sessions each.

Usage

```
fieldvoleCH
```

Format

The format is a multi-session secr capthist object. Attribute ‘ampm’ codes for type of secondary session (am, pm).

Details

Ergon and Lambin (2013) provided a robust design dataset from a trapping study on field voles *Microtus agrestis* in a clearcut within Kielder Forest, northern England – see also Ergon et al. (2011), Ergon and Gardner (2014) and Reich and Gardner (2014). The study aimed to describe sex differences in space-use, survival and dispersal among adult voles. Data were from one trapping grid in summer 2000.

Trapping was on a rectangular grid of 192 multi-catch (Ugglan Special) traps at 7-metre spacing. Traps were baited with whole barley grains and carrots; voles were marked with individually numbered ear tags.

Four trapping sessions were conducted at intervals of 21 to 23 days between 10 June and 15 August. Traps were checked at about 12 hour intervals (6 am and 6 pm).

The attribute 'ampm' is a data.frame with a vector of codes, one per secondary session, to separate am and pm trap checks (1 = evening, 2 = morning). The four primary sessions had respectively 3, 5, 4 and 5 trap checks.

Ergon and Gardner (2014) restricted their analysis to adult voles (118 females and 40 males). Histories of five voles (ma193, ma239, ma371, ma143, ma348) were censored part way through the study because they died in traps (T. Ergon pers. comm.).

Source

Data were retrieved from DRYAD (Ergon and Lambin (2013) for **openCR**. Code for translating the DRYAD ASCII file into a capthist object is given in Examples.

References

Efford, M. G. (2017) Multi-session models in secr 3.0. http://www.otago.ac.nz/density/pdfs/secr-multisession.pdf

Ergon, T., Ergon, R., Begon, M., Telfer, S. and Lambin, X. (2011) Delayed density- dependent onset of spring reproduction in a fluctuating population of field voles. *Oikos* **120**, 934–940.

Ergon, T. and Gardner, B. (2014) Separating mortality and emigration: modelling space use, dispersal and survival with robust-design spatial capture–recapture data. *Methods in Ecology and Evolution* **5**, 1327–1336.

Ergon, T. and Lambin, X. (2013) Data from: Separating mortality and emigration: Modelling space use, dispersal and survival with robust-design spatial capture–recapture data. Dryad Digital Repository. URL http://dx.doi.org/10.5061/dryad.r17n5.

Reich, B. J. and Gardner, B. (2014) A spatial capture–recapture model for territorial species. *Environmetrics* **25**, 630–637.

Examples

```
summary(fieldvoleCH, terse = TRUE)
m.array(fieldvoleCH)
JS.counts(fieldvoleCH)

maleCH <- subset(fieldvoleCH, function(x) covariates(x) == 'M')
fit <- openCR.fit(maleCH)
predict(fit)

attr(fieldvoleCH, 'ampm')

## Not run:

# Read data object from DRYAD ASCII file

datadir <- system.file('extdata', package = 'openCR')
EG <- dget(paste0(datadir,'/ergonandgardner2013.rdat'))
```

```
# construct capthist object
onesession <- function (sess) {
    mat <- EG$H[,,sess]
    id <- as.numeric(row(mat))
    occ <- as.numeric(col(mat))
    occ[mat<0] <- -occ[mat<0]
    trap <- abs(as.numeric(mat))
    matrow <- rownames(mat)
    df <- data.frame(session = rep(sess, length(id)),
                     ID = matrow[id],
                     occ = occ,
                     trapID = trap,
                     sex = c('F','M')[EG$gr],
                     row.names = 1:length(id))
    # retain captures (trap>0)
    df[df$trapID>0, , drop = FALSE]
}
tr <- read.traps(data = data.frame(EG$X), detector = "multi")

# recode matrix as mixture of zeros and trap numbers
EG$H <- EG$H-1

# code censored animals with negative trap number
# two ways to recognise censoring
censoredprimary <- which(EG$K < 4)
censoredsecondary <- which(apply(EG$J,1,function(x) any(x-c(3,5,4,5) < 0)))
censored <- unique(c(censoredprimary, censoredsecondary))
rownames(EG$H)[censored]
# [1] "ma193" "ma239" "ma371" "ma143" "ma348"
censorocc <- apply(EG$H[censored,,], 1, function(x) which.max(cumsum(x)))
censor3 <- ((censorocc-1) %/% 5)+1       # session
censor2 <- censorocc - (censor3-1) * 5   # occasion within session
censori <- cbind(censored, censor2, censor3)
EG$H[censori] <- -EG$H[censori]

lch <- lapply(1:4, onesession)
ch <- make.capthist(do.call(rbind,lch), tr=tr, covnames='sex')

# apply intervals in months
intervals(ch) <-  EG$dt

fieldvoleCH <- ch

# extract time covariate - each secondary session was either am (2) or pm (1)
# EG$tod
# 1 2 3  4  5
# 1 2 1 2 NA NA
# 2 2 1 2  1  1
# 3 2 1 2  1 NA
# 4 2 1 2  1  2
# Note consecutive pm trap checks in session 2
ampm <- split(EG$tod, 1:4)
ampm <- lapply(ampm, na.omit)
attr(fieldvoleCH, 'ampm') <- data.frame(ampm = unlist(ampm))

save(fieldvoleCH, file ='d:/open populations/openCR/data/fieldvole.RData')
```

```
## End(Not run)
```

---

gonodontisCH *Gonodontis Moths*

Description

Non-spatial open-population capture–recapture data of Bishop et al. (1978) for nonmelanic male *Gonodontis bidentata* at Cressington Park, northwest England.

Usage

```
gonodontisCH
```

Format

The format is a single-session secr capthist object. As these are non-spatial data, the traps attribute is NULL.

Details

The data are from a study of the relative fitness of melanic and nonmelanic morphs of the moth *Gonodontis bidentata* at several sites in England (Bishop et al. 1978). Crosbie (1979; see also Crosbie and Manly 1985) selected a subset of the Bishop et al. data (nonmelanic males from Cressington Park) to demonstrate innovations in Jolly-Seber modelling, and the same data were used by Link and Barker (2005) and Schofield and Barker (2008). The present data are those used by Crosbie (1979) and Link and Barker (2005).

Male moths were attracted to traps which consisted of a cage containing phermone-producing females surrounded by an enclosure which the males could enter but not leave. New virgin females were usually added every 1 to 4 days. Moths were marked at each capture with a date-specific mark in enamel paint or felt-tip pen on the undersurface of the wing. Thus, although moths at Cressington Park were not marked individually, each moth was a flying bearer of its own capture history.

The data comprise 689 individual capture histories for moths captured at 8 traps operated over 17 days (24 May–10 June 1970). The traps were in a square that appears have been about 40 m on a side. The location of captures is not included in the published data. All captured moths appear to have been marked and released (i.e. there were no removals recorded). All captures on Day 17 were recaptures; it is possible that unmarked moths were not recorded on that day.

Both Table 1 and Appendix 1 (microfiche) of Bishop et al. (1978) refer to 690 capture histories of nonmelanics at Cressington Park. In the present data there are only 689, and there are other minor discrepancies. Also, Crosbie and Manly (1985: Table 1) refer to 82 unique capture histories (“distinct cmr patterns”) when there are only 81 in the present dataset (note that two moths share 00000000000000011).

Source

Richard Barker provided an electronic copy of the data used by Link and Barker (2005), copied from Crosbie (1979).

References

Bishop, J. A., Cook, L. M., and Muggleton, J. (1978). The response of two species of moth to industrialization in northwest England. II. Relative fitness of morphs and population size. *Philosophical Transactions of the Royal Society of London* **B281**, 517–540.

Crosbie, S. F. (1979) *The mathematical modelling of capture–mark–recapture experiments on animal populations*. Ph.D. Thesis, University of Otago, Dunedin, New Zealand.

Crosbie, S. F. and Manly, B. F. J. (1985) Parsimonious modelling of capture–mark–recapture studies. *Biometrics* **41**, 385–398.

Link, W. A. and Barker, R. J. (2005) Modeling association among demographic parameters in analysis of open-population capture–recapture data. *Biometrics* **61**, 46–54.

Schofield, M. R. and Barker, R. J. (2008) A unified capture–recapture framework. *Journal of Agricultural Biological and Environmental Statistics* **13**, 458–477.

Examples

```
summary(gonodontisCH)
m.array(gonodontisCH)

## Not run:
# compare default (CJS) estimates from openCR, MARK

fit <- openCR.fit(gonodontisCH)
predict(fit)

library(RMark)
# may need to set MarkPath
mothdf <- RMarkInput(gonodontisCH)
mark(mothdf)
cleanup(ask = FALSE)

## End(Not run)
```

`Internal` *Internal Functions*

Description

Functions called by `openCR.fit` when `details$R == TRUE`

Usage

```
prwi(type, n, x, jj, cumss, nmix, w, fi, li, openval, PIA, PIAJ, intervals, CJSp1)

prwisecr(type, n, x, nc, jj, kk, mm, nmix, cumss, w, fi, li, gk, openval,
    PIA, PIAJ, binomN, Tsk, intervals, CJSp1, moveargsi, movemodel, usermodel,
    kernel = NULL, mqarray = NULL, cellsize = NULL)

PCH1(type, x, nc, cumss, nmix, openval0, PIA0, PIAJ, intervals)
```

```
PCH1secr(type, x, nc, jj, cumss, kk, mm, openval0, PIA0, PIAJ, gk0, binomN, Tsk,
    intervals,  moveargsi, movemodel, usermodel, kernel, mqarray, cellsize)

pradelloglik (type, w, openval, PIAJ, intervals)
```

Arguments

`type`	character
`n`	integer index of capture history
`x`	integer index of latent class
`jj`	integer number of primary sessions
`cumss`	integer vector cumulative number of secondary sessions at start of each primary session
`nmix`	integer number of latent classes
`w`	array of capture histories
`fi`	integer first primary session
`li`	integer last primary session
`openval`	dataframe of real parameter values (one unique combination per row)
`PIA`	parameter index array (secondary sessions)
`PIAJ`	parameter index array (primary sessions)
`intervals`	integer vector
`CJSp1`	logical; should CJS likelihood include first primary session?
`moveargsi`	integer 2-vector for index of move.a, move.b (negative if unused)
`movemodel`	character
`usermodel`	function to fill kernel
`kernel`	dataframe with columns x,y relative coordinates of kernel cell centres
`mqarray`	integer matrix
`cellsize`	numeric length of side of kernel cell
`gk`	real array
`Tsk`	array detector usage
`openval0`	openval for naive animals
`PIA0`	PIA for naive animals
`gk0`	gk for naive animals
`nc`	number of capture histories
`kk`	number of detectors
`mm`	number of points on habitat mask
`binomN`	code for distribution of counts (see `secr.fit`)

Details

To be completed

Value

Numeric components of likelihood.

See Also

openCR.fit

JS.counts *Summarise Non-spatial Open-population Data*

Description

Simple conventional summaries of data held in secr 'capthist' objects.

Usage

```
JS.counts(object, primary.only = TRUE)
m.array(object, primary.only = TRUE, never.recaptured = TRUE)
bd.array(beta, phi)
```

Arguments

`object`	secr capthist object or similar
`primary.only`	logical; if TRUE then counts are tabuated for primary sessions
`never.recaptured`	logical; if TRUE then a column is added for animals never recaptured
`beta`	numeric vector of entry probabilities, one per primary session
`phi`	numeric vector of survival probabilities, one per primary session

Details

The input is a capthist object representing a multi-session capture–recapture study. This may be (i) a single-session capthist in which occasions are understood to represent primary sessions, or (ii) a multi-session capthist object that is automatically converted to a single session object with `join` (any secondary sessions (occasions) are first collapsed with `reduce(object, by = 'all')`*).

The argument `primary.only` applies for single-session input with a robust-design structure defined by the `intervals`.

If the covariates attribute of `object` includes a column named 'freq' then this is used to expand the capture histories.

Conventional Jolly–Seber estimates may be computed with `JS.direct`.

`bd.array` computes the probability of each possible combination of birth and death times (strictly, the primary session at which an animal was first and last available for detection), given the parameter vectors beta and phi. These cell probabilities are integral to JSSA models.

* this may fail with nonspatial data.

Value

For `JS.counts`, a data.frame where rows correspond to sessions and columns hold counts as follows –

`n`	number of individuals detected
`R`	number of individuals released
`m`	number of previously marked individuals
`r`	number of released individuals detected in later sessions
`z`	number known to be alive (detected before and after) but not detected in current session

For `m.array`, a table object with rows corresponding to release cohorts and columns corresponding to first–recapture sessions. The size of the release cohort is shown in the first column. Cells in the lower triangle have value NA and print as blank by default.

See Also

`join`, `JS.direct`

Examples

```
JS.counts(ovenCH)
m.array(ovenCH)

## probabilities of b,d pairs
fit <- openCR.fit(ovenCH, type = 'JSSAbCL')
beta <- predict(fit)$b$estimate
phi <- predict(fit)$phi$estimate
bd.array(beta, phi)
```

`JS.direct` *Jolly–Seber Estimates*

Description

Non-spatial open-population estimates using the conventional closed-form Jolly–Seber estimators (Pollock et al. 1990).

Usage

```
JS.direct(object)
```

Arguments

`object`	secr capthist object or similar

Details

Estimates are the session-specific Jolly-Seber estimates with no constraints.

The reported SE of births (B) differ slightly from those in Pollock et al. (1990), and may be in error.

Value

A dataframe in which the first 5 columns are summary statistics (counts from `JS.counts`) and the remaining columns are estimates:

`p`	capture probability
`N`	population size
`phi`	probability of survival to next sample time
`B`	number of recruits at next sample time

Standard errors are in fields prefixed 'se'; for N and B these include only sampling variation and omit population stochasticity. The covariance of successive phi-hat is in the field 'covphi'.

References

Pollock, K. H., Nichols, J. D., Brownie, C. and Hines, J. E. (1990) Statistical inference for capture–recapture experiments. *Wildlife Monographs* **107**. 97pp.

See Also

`JS.counts`

Examples

```
# cf Pollock et al. (1990) Table 4.8
JS.direct(microtusCH)
```

`LLsurface` *Plot Likelihood Surface*

Description

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

This is a method for the generic function `LLsurface` defined in **secr**.

Usage

```
## S3 method for class 'openCR'
LLsurface(object, betapar = c("phi", "sigma"), xval = NULL, yval = NULL,
  centre = NULL, realscale = TRUE, plot = TRUE, plotfitted = TRUE, ncores = 1, ...)
```

Arguments

`object`	openCR object output from `openCR.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 (+)
`ncores`	integer number of cores available for parallel processing
`...`	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.

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

Value

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

Note

`LLsurface.openCR` 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 Also

`LLsurface.secr`

Examples

```
# not yet
```

make.table *Tabulate Estimates From Multiple Models*

Description

Session-specific estimates of real parameters (p, phi, etc.) are arranged in a rectangular table.

Usage

```
make.table(fits, parm = "phi", fields = "estimate", ...)
```

Arguments

`fits`	openCRlist object
`parm`	character name of real parameter estimate to tabulate
`fields`	character column from predict (estimate, SE.estimate, lcl, ucl)
`...`	arguments passed to `predict.openCRlist`

Details

The input will usually be from `par.openCR.fit`.

Value

A table object.

See Also

`par.openCR.fit`, `openCRlist`

Examples

```
arglist <- list(constant = list(capthist=ovenCHp, model=phi~1),
                session.specific = list(capthist=ovenCHp, model=phi~session))
fits <- par.openCR.fit(arglist, trace = FALSE)
print(make.table(fits), na=".")
```

`Microtus` *Patuxent Meadow Voles*

Description

Captures of *Microtus pennsylvanicus* at Patuxent Wildlife Research Center, Laurel, Maryland, June–December 1981. Collapsed (primary session only) data for adult males and adult females, and full robust-design data for adult males. Nichols et al. (1984) described the field methods and analysed a superset of the present data.

Usage

```
microtusCH
microtusFCH
microtusMCH
microtusFMCH
microtusRDCH
```

Format

The format is a single-session secr capthist object. As these are non-spatial data, the traps attribute is NULL.

Details

Voles were caught in live traps on a 10 x 10 grid with traps 7.6 m apart. Traps were baited with corn. Traps were set in the evening, checked the following morning, and locked open during the day. Voles were ear-tagged with individually numbered fingerling tags. The locations of captures were not included in the published data.

Data collection followed Pollock's robust design with five consecutive days of trapping each month for six months (27 June 1981–8 December 1981). The data are for "adult" animals only, defined as those weighing at least 22g. Low capture numbers on the last two days of the second primary session (occasions 9 and 10) are due to a raccoon interfering with traps (Nichols et al. 1984). Six adult female voles and ten adult male voles were not released; their final captures are coded as -1 in the respective capthist objects.

`microtusRDCH` is the full robust-design dataset for adult males ((Williams et al. 2002 Table 19.1).

`microtusFCH` and `microtusMCH` are the collapsed datasets (binary at the level of primary session) for adult females and adult males from Williams et al. (2002 Table 17.5); `microtusFMCH` combines them and includes the covariate 'sex'.

`microtusCH` is a combined-sex version of the data with different lineage (see below).

The 'intervals' attribute was assigned for `microtusRDCH` to distinguish primary sesssions (interval 1 between prmary sessions; interval 0 for consecutive secondary sessions within a primary session). True intervals (start of one primary session to start of next) were 35, 28, 35, 28 and 34 days. See Examples to add these manually.

Williams, Nichols and Conroy (2002) presented several analyses of these data.

Program JOLLY (Hines 1988, Pollock et al. 1990) included a combined-sex version of the primary-session data that was used by Pollock et al. (1985) and Pollock et al. (1990)*. The numbers of voles released each month in the JOLLY dataset JLYEXMPL differ by 0–3 from the sum of the male and female data from Williams et al. (2002) (see Examples). Some discrepancies may have been due to voles for which sex was not recorded. The JOLLY version matches Table 1 of Nichols et al. (1984). The JOLLY version is distributed here as the object `microtusCH`.

Differing selections of data from the Patuxent study were analysed by Nichols et al. (1992) and Bonner and Schwarz (2006).

* There is a typographic error in Table 4.7 of Pollock et al. (1990): r_i for the first period should be 89.

Source

Object	Source
`microtusCH`	Text file JLYEXMPL distributed with Program JOLLY (Hines 1988; see also Examples)
`microtusFCH`	Table 17.5 in Williams, Nichols and Conroy (2002)
`microtusMCH`	Table 17.5 in Williams, Nichols and Conroy (2002)
`microtusFMCH`	Table 17.5 in Williams, Nichols and Conroy (2002)
`microtusRDCH`	Table 19.1 in Williams, Nichols and Conroy (2002) provided as text file by Jim Hines

References

Bonner, S. J. and Schwarz, C. J. (2006) An extension of the Cormack–Jolly–Seber model for continuous covariates with application to *Microtus pennsylvanicus*. *Biometrics* **62**, 142–149.

Hines, J. E. (1988) Program "JOLLY". Patuxent Wildlife Research Center. https://www.mbr-pwrc.usgs.gov/software/jolly.shtml

Nichols, J. D., Pollock, K. H., Hines, J. E. (1984) The use of a robust capture-recapture design in small mammal population studies: a field example with *Microtus pennsylvanicus*. *Acta Theriologica* **29**, 357–365.

Nichols, J. D., Sauer, J. R., Pollock, K. H., and Hestbeck, J. B. (1992) Estimating transition probabilities for stage-based population projection matrices using capture–recapture data. *Ecology* **73**, 306–312.

Pollock, K. H., Hines, J. E. and Nichols, J. D. (1985) Goodness-of-fit tests for open capture–recapture models. *Biometrics* **41**, 399–410.

Pollock, K. H., Nichols, J. D., Brownie, C. and Hines, J. E. (1990) Statistical inference for capture–recapture experiments. *Wildlife Monographs* **107**. 97pp.

Williams, B. K., Nichols, J. D. and Conroy, M. J. (2002) *Analysis and management of animal populations.* Academic Press.

Examples

```
# cf Williams, Nichols and Conroy Table 17.6
m.array(microtusFCH)
m.array(microtusMCH)

# cf Williams, Nichols and Conroy Fig. 17.2
fitfm <- openCR.fit(microtusFMCH, model = list(p~1, phi ~ session + sex))
maledat <- expand.grid(sex = factor('M', levels = c('F','M')), session = factor(1:6))
```

```
plot(fitfm, ylim=c(0,1), type = 'o')
plot(fitfm, newdata = maledat, add = TRUE, xoffset = 0.1, pch = 16, type = 'o')

# adjusting for variable interval
intervals(microtusCH) <-  c(35,28,35,28,34) / 30
intervals(microtusRDCH)[intervals(microtusRDCH)>0] <- c(35,28,35,28,34) / 30

## Not run:
# The text file JLYEXMPL distributed with JOLLY is in the extdata folder of the R package
# The microtusCH object may be rebuilt as follows
datadir <- system.file('extdata', package = 'openCR')
JLYdf <- read.table(paste0(datadir,'/JLYEXMPL'), skip = 3,
                    colClasses = c('character','numeric'))
names(JLYdf) <- c('ch', 'freq')
JLYdf$freq[grepl('2', JLYdf$ch)] <- -JLYdf$freq[grepl('2', JLYdf$ch)]
JLYdf$ch <- gsub ('2','1', JLYdf$ch)
microtusCH <- unRMarkInput(JLYdf)

# Compare to combined-sex data from Williams et al. Table 17.5
JS.counts(microtusCH) - JS.counts(microtusFMCH)

## End(Not run)
```

---

miscellaneous *Data Manipulation*

Description

Miscellaneous functions

Usage

```
primarysessions(intervals)
secondarysessions(intervals)
timevaryingcov(object, ...)
timevaryingcov(object) <- value
```

Arguments

intervals	numeric vector of intervals for time between secondary sessions a of robust design
object	single-session capthist object
value	a list of named vectors
...	other arguments (not used)

Details

These functions are used internally.

Value

primarysessions –

Integer vector with the number of the primary session to which each secondary session belongs.

secondarysessions –

Integer vector with secondary sessions numbered sequentially within primary sessions.

Examples

```
int <- intervals(join(ovenCH))
primary <- primarysessions(int)
primary

# number of secondary sessions per primary
table(primary)

# secondary session numbers
secondarysessions(int)
```

---

openCR.design *Design Data for Open population Models*

Description

Internal function used by openCR.fit.

Usage

```
openCR.design(capthist, models, type, naive = FALSE,
    timecov = NULL, sessioncov = NULL, dframe = NULL,
    contrasts = NULL, initialage = 0, maximumage = 1, ...)
```

Arguments

capthist	single-session capthist object
models	list of formulae for parameters of detection
type	character string for type of analysis "CJS", "JSSA" or "Pradel"
naive	logical if TRUE then modelled parameter is for a naive animal (not caught previously)
timecov	optional vector or dataframe of values of occasion-specific covariate(s).
sessioncov	optional dataframe of values of session-specific covariate(s)
dframe	optional data frame of design data for detection parameters
contrasts	contrast specification as for model.matrix
initialage	numeric or character (name of individual covariate containing initial ages)
maximumage	numeric; age at which to truncate
...	other arguments passed to the R function model.matrix

Details

This is an internal **openCR** 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 individual, occasion and latent class, i.e., for n individuals, J occasions and m latent classes there are *potentially* $n \times J \times m$ 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.

`openCR.fit` is more tolerant than `openCR.design` regarding the inputs 'capthist' and 'models'. Model formulae are processed to a standard form (a named list of formulae) before they are passed to `openCR.design`, and multi-session capthist objects are automatically 'reduced' and 'joined' for open-population analysis.

If `timecov` is a single vector of values (one for each secondary session) then it is treated as a covariate named 'tcov'. If `sessioncov` is a single vector of values (one for each primary session) then it is treated as a covariate named 'scov'.

The `initialage` and `maximumage` arguments are usually passed via the `openCR.fit` 'details' argument.

Value

A list with the components

`designMatrices`	list of reduced design matrices, one for each real parameter
`parameterTable`	index to row of the reduced design matrix for each real parameter; dim(parameterTable) = c(uniquepar, np), where uniquepar is the number of unique combinations of paramater values (uniquepar < nJM) and np is the number of parameters in the detection model.
`PIA`	Parameter Index Array - index to row of parameterTable for a given animal, occasion and latent class; dim(PIA) = c(n,J,M)
`validlevels`	a logical matrix of np rows and J columns, mostly TRUE, but FALSE for impossible combinations e.g. CJS recapture probability in session 1 (validlevels["p",1]), or CJS final survival probability (validlevels["phi",J]). Also, validlevels["b",1] is FALSE with type = "JSSA" because of the constraint that entry parameters sum to one.

Note

The component validlevels is TRUE in many cases for which a parameter is redundant or confounded (e.g. validlevels["phi",J-1]); these are sorted out 'post hoc' by examining the fitted values, their asymptotic variances and the eigenvalues of the Hessian matrix.

See Also

`openCR.fit`

Examples

```
## this happens automatically in openCR.fit
ovenCH1 <- join(reduce(ovenCH, by = "all", newtraps=list(1:44)))
```

```
openCR.design (ovenCH1, models = list(p = ~1, phi = ~session),
    interval = c(1,1,1,1), type = "CJS")
```

---

openCR.fit *Fit Open Population Capture–Recapture Model*

Description

Nonspatial or spatial open-population analyses are performed on data formatted for 'secr'. Several parameterisations are provided for the nonspatial Jolly-Seber Schwarz-Arnason model ('JSSA', also known as 'POPAN'). Corresponding spatial models are designated 'JSSAsecr'. Cormack-Jolly-Seber (CJS) models are also fitted.

Usage

```
openCR.fit (capthist, type = "CJS", model = list(p~1, phi~1, sigma~1),
    distribution = c("poisson", "binomial"), mask = NULL,
    detectfn = c("HHN","HHR","HEX","HAN","HCG","HVP"),
   binomN = 0, movementmodel = c("static", "uncorrelated", "normal", "exponential"),
    start = NULL, link = list(), fixed = list(), timecov = NULL,
    sessioncov = NULL, dframe = NULL, details = list(),
    method = "Newton-Raphson", trace = NULL, ncores = 1, ...)
```

Arguments

`capthist`	`capthist` object from 'secr'
`type`	character string for type of analysis (see Details)
`model`	list with optional components, each symbolically defining a linear predictor for the relevant real parameter using `formula` notation. See Details for names of real parameters.
`distribution`	character distribution of number of individuals detected
`mask`	single-session `mask` object; required for spatial (secr) models
`detectfn`	character code
`binomN`	integer code for distribution of counts (see `secr.fit`)
`movementmodel`	character; model for movement between primary sessions (see Details)
`start`	vector of initial values for beta parameters, or fitted model(s) from which they may be derived
`link`	list with named components, each a character string in {"log", "logit", "loglog", identity", "sin", "mlogit"} for the link function of the relevant real parameter
`fixed`	list with optional components corresponding to each 'real' parameter, the scalar value to which parameter is to be fixed
`timecov`	optional dataframe of values of occasion-specific covariate(s).
`sessioncov`	optional dataframe of values of session-specific covariate(s).
`dframe`	optional data frame of design data for detection parameters (seldom used)
`details`	list of additional settings (see Details)

`method`	character string giving method for maximizing log likelihood
`trace`	logical, if TRUE then output each evaluation of the likelihood, and other messages
`ncores`	integer number of cores for parallel processing
`...`	other arguments passed to join()

Details

The permitted nonspatial models are CJS, Pradel, Pradelg, JSSAbCL, JSSAfCL, JSSAgCL, JSSAlCL, JSSAb, JSSAf, JSSAg, JSSAl, JSSAB and JSSAN. The permitted spatial models are CJSsecr, JSSAsecrbCL, JSSAsecrfCL, JSSAsecrgCL, JSSAsecrlCL, JSSAsecrb, JSSAsecrf, JSSAsecrg, JSSAsecrl, JSSAsecrB and JSSAsecrN. See the openCR-vignette.pdf for a table of the 'real' parameters associated with each model type.

Parameterisations of the JSSA models differ in how they include recruitment: the core parameterisations express recruitment either as a per capita rate ('f'), as a finite rate of increase for the population ('l' for lambda) or as per-occasion entry probability ('b' for the classic JSSA beta parameter, aka PENT in MARK). Each of these models may be fitted by maximising either the full likelihood, or the likelihood conditional on capture in the Huggins (1989) sense, distinguished by the suffix 'CL'. Full-likelihood JSSA models may also be parameterized in terms of the time-specific absolute recruitment (BN, BD) or the time-specific population size(N) or density (D).

Data are provided as **secr** 'capthist' objects, with some restrictions. For nonspatial analyses, 'capthist' may be single-session or multi-session, with any of the main detector types. For spatial analyses 'capthist' should be a single-session dataset of a point detector type ('multi', 'proximity' or 'count') (see also details$distribution below). In openCR the occasions of a single-session dataset are treated as open-population temporal samples except that occasions separated by an interval of zero (0) are from the same primary session (multi-session input is collapsed to single-session if necessary).

`model` formulae may include the pre-defined terms 'b', 'B', 'session','Session', 'h2', and 'h3' as in **secr**. 'session' is the name given to primary sampling times in 'secr', so a fully time-specific CJS model is `list(p ~ session, phi ~ session)`. 'b' refers to a within-session (learned) response to capture and 'B' to a transient (Markovian) response. 'bsession' is used for a multi-session learned response.'Session' is for a trend over sessions. 'h2' and 'h3' allow finite mixture models. Formulae may also include named occasion-specific and session-specific covariates in the dataframe arguments 'timecov' and 'sessioncov' (occasion = secondary session of robust design). Individual covariates present as an attribute of the 'capthist' input may be used in CJS and ..CL models. Groups are not supported in this version, but may be implemented via a factor-level covariate in ..CL models.

`distribution` specifies the distribution of the number of individuals detected; this may be conditional on the population size (or number in the masked area) ("binomial") or unconditional ("poisson"). `distribution` affects the sampling variance of the estimated density. The default is "binomial". For variance comparable with **secr** estimates this should be changed to "poisson".

[Movement models are under development]

The mlogit link function is used for the JSSA (POPAN) entry parameter 'b' (PENT in MARK) and for mixture proportions, regardless of `link`.

Spatial models use one of the hazard-based detection functions and require data from independent point detectors (**secr** detector types 'multi', 'proximity' or 'count').

The ... argument may be used to pass a vector of unequal intervals to join (`interval`), or to vary the tolerance for merging detector sites (`tol`).

The `start` argument may be

- a vector of beta parameter values, one for each of the NP beta parameters in the model

- a named vector of beta parameter values in any order

- a named list of one or more real parameter values

- a single fitted secr or openCR model whose real parameters overlap with the current model

- a list of two fitted models

In the case of two fitted models, the values are melded. This is handy for initialising an open spatial model from a closed spatial model and an open non-spatial model. If a beta parameter appears in both models then the first is used.

`details` is used for various specialized settings –

`details$autoini` (default 1) is the number of the session used to determine initial values of D, lambda0 and sigma (secr types only).

`details$contrasts` may be used to specify the coding of factor predictors. The value should be suitable for the 'contrasts.arg' argument of `model.matrix`.

`details$control` is a list that is passed to `optim` - useful for increasing maxit for `method = Nelder-Mead` (see vignette).

`details$fixedbeta` may be a vector with one element for each coefficient (beta parameter) in the model. Only 'NA' coefficients will be estimated; others will be fixed at the value given (coefficients define a linear predictor on the link scale). The number and order of coefficients may be determined by calling `openCR.fit` with trace = TRUE and interrupting execution after the first likelihood evaluation.

`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$initialage` is either numeric (the uniform age at first capture) or a character value naming an individual covariate with initial ages; see `age.matrix`.

`details$LLonly` = TRUE causes the function to returns a single evaluation of the log likelihood at the initial values, followed by the initial values.

`details$maximumage` sets a maximum age; older animals are recycled into this age class; see `age.matrix`.

`details$multinom` = TRUE includes the multinomial constant in the reported log-likelihood (default FALSE).

`details$R == TRUE` may be used to switch from the default C++ code to slower functions in native R (useful mostly for debugging; not all model types implemented).

`details$squeeze == TRUE` (the default) compacts the input capthist with function `squeeze` before analysis. The new capthist includes only unique rows. Non-spatial models will fit faster, because non-spatial capture histories are often non-unique.

If `method = "Newton-Raphson"` then `nlm` is used to maximize the log likelihood (minimize the negative log likelihood); otherwise `optim` is used with the chosen method ("BFGS", "Nelder-Mead", etc.). If maximization fails a warning is given appropriate to the method. `method = "none"` may be used to compute or re-compute the variance-covariance matrix at given starting values (i.e. providing a previously fitted model as the value of `start`).

Parameter redundancies are common in open-population models. The output from `openCR.fit` includes the singular values (eigenvalues) of the Hessian - a useful post-hoc indicator of redundancy (e.g., Gimenez et al. 2004). Eigenvalues are scaled so the largest is 1.0. Very small scaled values represent redundant parameters - in my experience with simple JSSA models a threshold of 0.00001 seems effective.

[There is an undocumented option to fix specific ‘beta’ parameters.]

Value

If `details$LLonly == TRUE` then a numeric vector is returned with logLik in position 1, followed by the named coefficients.

Otherwise, an object of class ‘openCR’ with components

model = model, distribution = distribution, mask = mask, detectfn = detectfn, binomN = binomN, movementmodel = movementmodel, usermodel = usermodel, moveargsi = moveargsi, start = start,

`call`	function call
`capthist`	saved input
`type`	saved input
`model`	saved input
`distribution`	saved input
`mask`	saved input
`detectfn`	saved input
`binomN`	saved input
`movementmodel`	saved input
`usermodel`	saved input
`moveargsi`	relative locations of move.a and move.b arguments
`start`	vector of starting values for beta parameters
`link`	saved input
`fixed`	saved input
`timecov`	saved input
`sessioncov`	saved input
`dframe`	saved input
`details`	saved input
`method`	saved input
`ncores`	saved input
`design`	reduced design matrices, parameter table and parameter index array for actual animals (see `openCR.design`)
`design0`	reduced design matrices, parameter table and parameter index array for ‘naive’ animal (see `openCR.design`)
`parindx`	list with one component for each real parameter giving the indices of the ‘beta’ parameters associated with each real parameter
`intervals`	intervals between primary sessions
`vars`	vector of unique variable names in `model`
`betanames`	names of beta parameters
`realnames`	names of fitted (real) parameters
`sessionlabels`	name of each primary session

`fit`	list describing the fit (output from `nlm` or `optim`)
`beta.vcv`	variance-covariance matrix of beta parameters
`eigH`	vector of eigenvalue corresponding to each beta parameter
`posterior`	posterior probabilities of class membership (mixture models), one row per individual.
`version`	openCR version number
`starttime`	character string of date and time at start of fit
`proctime`	processor time for model fit, in seconds

Note

Different parameterisations lead to different model fits when used with the default 'model' argument in which each real parameter is constrained to be constant over time.

The JSSA implementation uses summation over feasible 'birth' and 'death' times for each capture history, following Pledger et al. (2010). This enables finite mixture models for individual capture probability (not fully tested), flexible handling of additions and losses on capture (aka removals) (not yet programmed), and ultimately the extension to 'unknown age' as in Pledger et al. (2009).

openCR uses the generalized matrix inverse 'ginv' from the MASS package rather than 'solve' from base R, as this seems more robust to singularities in the Hessian. Also, the default maximization method is 'BFGS' rather than 'Newton-Raphson' as BFGS appears more robust in the presence of redundant parameters.

References

Gimenez, O., Viallefont, A., Catchpole, E. A., Choquet, R. and Morgan, B. J. T. (2004) Methods for investigating parameter redundancy. *Animal Biodiversity and Conservation* **27**, 561–572.

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

Pledger, S., Efford, M., Pollock. K., Collazo, J. and Lyons, J. (2009) Stopover duration analysis with departure probability dependent on unknown time since arrival. In: D. L. Thompson, E. G. Cooch and M. J. Conroy (eds) *Modeling Demographic Processes in Marked Populations*. Springer. Pp. 349–363.

Pledger, S., Pollock, K. H. and Norris, J. L. (2010) Open capture–recapture models with heterogeneity: II. Jolly-Seber model. *Biometrics* **66**, 883–890.

Pradel, R. (1996) Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* **52**, 703–709.

Schwarz, C. J. and Arnason, A. N. (1996) A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics* **52**, 860–873.

See Also

`openCR.design`, `derived.openCR`, `par.openCR.fit`

Examples

```
## CJS default
openCR.fit(ovenCH)

## POPAN Jolly-Seber Schwarz-Arnason, lambda parameterisation
L1 <- openCR.fit(ovenCH, type = 'JSSAl')
```

```
L1CL <- openCR.fit(ovenCH, type = 'JSSAlCL')
predict(L1)
predict(L1CL)

## Not run:
JSSA1 <- openCR.fit(ovenCH, type = 'JSSAf')
JSSA2 <- openCR.fit(ovenCH, type = 'JSSAf', model = list(phi~t))
JSSA3 <- openCR.fit(ovenCH, type = 'JSSAf', model = list(p~t,phi~t))
AIC (JSSA1, JSSA2, JSSA3)
predict(JSSA1)

RMdata <- RMarkInput (join(reduce(ovenCH, by = "all")))
require(RMark)
openCHtest <- process.data(RMdata, model='POPAN')
openCHPOPAN <- mark(data = openCHtest, model = 'POPAN',
    model.parameters = list(p = list(formula = ~1),
    pent = list(formula = ~1),
    Phi = list(formula = ~1)))
popan.derived(openCHtest, openCHPOPAN)
cleanup(ask=F)

## End(Not run)
```

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

Description

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

Usage

```
openCR.make.newdata(object, all.levels = FALSE)
```

Arguments

`object`	fitted openCR model object
`all.levels`	logical; if TRUE then all covariate factor levels appear in the output

Details

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

Value

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

See Also

openCR.fit

Examples

```
## null example (no covariates)
ovenCJS <- openCR.fit(ovenCH)
openCR.make.newdata(ovenCJS)
```

---

par.openCR.fit *Fit Multiple openCR Models*

Description

This function is a wrappers for openCR.fit.

Usage

```
par.openCR.fit (arglist, ncores = 1, seed = 123, trace = TRUE, logfile = "logfile.txt",
    prefix = "")

openCRlist (...)
```

Arguments

arglist	list of argument lists for secr.fit or a character vector naming such lists
ncores	integer number of cores to be used for parallel processing
seed	integer pseudorandom number seed
trace	logical; if TRUE intermediate output may be logged
logfile	character name of file to log progress reports
prefix	character prefix for names of output
...	openCR objects

Details

Any attempt in arglist to set ncores > 1 for a particular secr fit is ignored.

trace overrides any settings in arglist. Reporting of intermediate results is unreliable on Windows when ncores > 1.

It is convenient to provide the names of the capthist and mask arguments in each component of arglist as character values (i.e. in quotes); objects thus named are exported from the workspace to each worker process (see Examples).

openCRlist forms a special list (class openCRlist) of fitted model (openCR) objects.

Value

For `par.openCR.fit` - openCRlist of model fits (see `openCR.fit`). Names are created by prefixing `prefix` to the names of `argslist`. If `trace` is TRUE then the total execution time and finish time are displayed.

See Also

`openCR.fit`, Parallel, `make.table`

Examples

```
## Not run:

m1 <- list(capthist = ovenCH, model = list(p~1, phi~1))
m2 <- list(capthist = ovenCH, model = list(p~session, phi~1))
m3 <- list(capthist = ovenCH, model = list(p~session, phi~session) )
fits <- par.openCR.fit (c('m1','m2','m3'), ncores = 3)
AIC(fits)


## End(Not run)
```

`plot.openCR` *Plot Estimates*

Description

Session-specific estimates of the chosen parameter are plotted.

Usage

```
## S3 method for class 'openCR'
plot(x, par = "phi", newdata = NULL, add = FALSE, xoffset = 0, ylim = NULL,
   useintervals = TRUE, CI = TRUE, intermediate.x = TRUE, alpha = 0.05, ...)
```

Arguments

`x`	openCR object from openCR.fit
`par`	character names of parameter to plot
`newdata`	dataframe of predictor values for `predict` (optional)
`add`	logical; if TRUE then points are added to an existing plot
`xoffset`	numeric offset to be added to all x values
`ylim`	numeric vector of limits on y-axis
`useintervals`	logical; if TRUE then x values are spaced according to the intervals attribute
`CI`	logical; if TRUE then 1-alpha confidence intervals are plotted
`intermediate.x`	logical; if TRUE then turnover parameters are plotted at the mid point on the x axis of the interval to which they relate
`alpha`	numeric confidence level default (alpha = 0.05) is 95% interval
`...`	other arguments passed to `points`

Details

If ylim is not provided it is set automatically.

Note that the ... argument is passed only to `points`. If you wish to customize the base plot then do that in advance and use `add = TRUE`.

Value

None

See Also

`predict`

Examples

```
## Not run:

fit <- openCR.fit(join(ovenCH), type='CJS', model = phi~session)
plot(fit,'phi', pch = 16, cex=1.3, yl=c(0,1))


## End(Not run)
```

`plotKernel` *Plot Movement Kernel*

Description

Movement between primary sessions is modelled in **openCR** with a discretized kernel. Each cell of the kernel contains the probability of movement from the central cell. Kernels are 'normal' (Gaussian), 'exponential' (negative exponential) or specified with a user-provided function. This function allows you to preview a kernel specification.

Usage

```
plotKernel(movementmodel = c("normal", "exponential"), kernelradius = 10, spacing, pars,
    clip = FALSE, plt = TRUE, contour = FALSE, levels = NULL, text = FALSE, ...)
```

Arguments

`movementmodel`	character or function
`kernelradius`	integer radius of kernel in grid cells
`spacing`	numeric spacing between cell centres
`pars`	numeric vector of 1 or 2 parameter values
`clip`	logical; if TRUE then corner cells are removed
`plt`	logical; if TRUE then a plot is produced

`contour`	logical; if TRUE then contour lines are overlaid on any plot
`levels`	numeric vector of contour levels
`text`	logical; if TRUE then cell probabilities are overprinted, roumnded to 3 d.p.
`...`	other arguments passed to plot.mask (e.g. breaks)

Details

Internally, a mask is generated with kernel probabilities in a covariate, and plotting is done with `plot.mask`.

Value

A dataframe with columns x, y, and kernelp is returned invisibly.

Examples

```
plotKernel(spacing = 2, k = 10, pars = 10, contour = TRUE, clip = TRUE)
```

`PPNpossums` *Orongorongo Valley Brushtail Possums*

Description

A subset of brushtail possum (*Trichosurus vulpecula*) data from the Orongorongo Valley live-trapping study of Efford (1998) and Efford and Cowan (2005) that was used by Pledger, Pollock and Norris (2003, 2010). The `OVpossumCH` dataset in **secr** is a different selection of data from the same study. Consult ?OVpossumCH for more detail.

The data comprise captures in February of each year from 1980 to 1988.

Usage

```
FebpossumCH
```

Format

The format is a 9-session **secr** capthist object. Capture locations are not included.

Details

The data are captures of 448 animals (175 females and 273 males) over 9 trapping sessions comprising 4–10 occasions each. All were independent of their mothers, but age was not otherwise distinguished. The individual covariate `sex` takes values 'F' or 'M'.

Pledger, Pollock and Norris (2010) fitted 2-class finite mixture models for capture probability p and apparent survival phi, with or without allowance for temporal (between year) variation, using captures from only the first day of each trapping session. The first-day data relate to 270 individuals (115 females and 155 males).

Source

M. Efford unpubl. See Efford and Cowan (2004) for acknowledgements.

References

Efford, M. G. (1998) Demographic consequences of sex-biased dispersal in a population of brushtail possums. *Journal of Animal Ecology* **67**, 503–517.

Efford, M. G. and Cowan, P. E. (2004) Long-term population trend of *Trichosurus vulpecula* in the Orongorongo Valley, New Zealand. In: *The Biology of Australian Possums and Gliders*. Edited by R. L. Goldingay and S. M. Jackson. Surrey Beatty & Sons, Chipping Norton. Pp. 471–483.

Pledger, S., Pollock, K. H. and Norris, J. L. (2010) Open capture–recapture models with heterogeneity: II. Jolly–Seber model. *Biometrics* **66**, 883–890.

Examples

```
summary(FebpossumCH)
m.array(FebpossumCH)
JS.counts(FebpossumCH)

FebD1CH <- subset(FebpossumCH, occasion = 1)

## Not run:

# reading the text file 'poss8088.data'

datadir <- system.file('extdata', package = 'openCR')
poss8088df <- read.table (paste0(datadir,'/poss8088.data'), header = TRUE)
capt <- poss8088df[,c('session','id','day','day','sex')]

# duplication of day is a trick to get a dummy trapID column in the right place
# this is needed because make.capthist does not have nonspatial option
capt$day.1[] <- 1

# keep only February samples
capt <- capt[capt$session %% 3 == 1,]

# build nonspatial secr capthist object using dummy trapping grid
FebpossumCH <- make.capthist(capt, make.grid(1,2,ID='numx'))
# discard dummy traps objects
for (i in 1:9) attr(FebpossumCH[[i]], 'traps') <- NULL
names(FebpossumCH) <- 1980:1988
sessionlabels(FebpossumCH) <- 1980:1988


## End(Not run)
```

`predict.openCR` *openCR Model Predictions*

Description

Evaluate an openCR 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 'openCR'
 predict(object, newdata = NULL, se.fit = TRUE, alpha = 0.05, savenew = FALSE, ...)
## S3 method for class 'openCRlist'
 predict(object, newdata = NULL, se.fit = TRUE, alpha = 0.05, savenew = FALSE, ...)
```

Arguments

`object`	openCR object output from `openCR.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
`savenew`	logical; if TRUE then newdata is saved as an attribute
`...`	other arguments passed to `openCR.make.newdata`

Details

Predictions are provided for each row in 'newdata'. The default (constructed by `openCR.make.newdata`) is to limit those rows to the first-used level of factor predictors; to include all levels pass `all.levels = TRUE` to `openCR.make.newdata` in the ... argument.

See Also

`AIC.openCR`, `openCR.fit`

Examples

```
c1 <- openCR.fit(ovenCH, type='CJS', model=phi~session)
predict(c1)
```

`print.derivedopenCR` *Print Method for Derived Estimates*

Description

Formats output from `derived.openCR`.

Usage

```
## S3 method for class 'derivedopenCR'
print(x, Dscale = NULL, legend = FALSE, ...)
```

Arguments

`x`	object from `derived.openCR`
`Dscale`	numeric optional multiplier for densities (overrides saved Dscale)
`legend`	logical. if TRUE then a legend is provided to column headings
`...`	other arguments passed to `print.data.frame`

Details

By default (i.e. when not not specified in the in the ... argument), `row.names = FALSE` and `digits = 4`.

See Also

`derived.openCR`

`print.openCR` *Print openCR Object*

Description

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

Usage

```
## S3 method for class 'openCR'
 print(x, newdata = NULL, alpha = 0.05, svtol = 1e-5,...)
```

Arguments

`x`	openCR object output from `openCR.fit`
`newdata`	optional dataframe of values at which to evaluate model
`alpha`	alpha level
`svtol`	threshold for non-null eigenvalues when computing numerical rank
`...`	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 fitting started
N animals	number of distinct animals detected
N captures	number of detections
N sessions	number of sampling occasions
Model	model formula for each 'real' parameter

Fixed	fixed real parameters
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
Eigenvalues	scaled eigenvalues of Hessian matrix (maximum 1.0)
Numerical rank	number of eigenvalues exceeding svtol
vcov	variance-covariance matrix of beta parameters
Real parameters	fitted (real) parameters evaluated at base levels of covariates

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.openCR`, `openCR.fit`

Examples

```
c1 <- openCR.fit(ovenCH, type='CJS', model=phi~session)
c1
```

---

read.inp — *Import Data from RMark Input Format*

Description

`read.inp` forms a capthist object from a MARK input (.inp) file.

Usage

```
read.inp(filename, ngroups = 1, grouplabel = 'group', grouplevels = NULL,
    covnames = NULL, skip = 0)
```

Arguments

`filename`	character file name including '.inp'.
`ngroups`	integer number of group columns in input
`grouplabel`	character
`grouplevels`	vector with length equal to number of groups
`covnames`	character vector of additional covariates names, one per covariate column
`skip`	integer number of lines to skip at start of file

Details

Comments bracketed with '/*' and '*/' will be removed automatically.

If grouplevels is specified then ngroups is taken from the number of levels (ngroups is overridden). An individual covariate is output, named according to grouplabel. The order of levels in grouplevels should match the order of the group frequency columns in the input. This also determines the ordering of levels in the resulting covariate.

Value

A single-session capthist object with no traps attribute.

See Also

RMarkInput, unRMarkInput

Examples

```
datadir <- system.file('extdata', package = 'openCR')
dipperCH <- read.inp(paste0(datadir, '/ed.inp'), ngroups = 2)
summary(dipperCH)
```

---

simulation *Simulate Capture Histories*

Description

Generate non-spatial or spatial open-population data and fit models.

Usage

```
sim.nonspatial (N, turnover = list(), p, nsessions, noccasions = 1, intervals = NULL,
    recapfactor = 1, seed = NULL, savepopn = FALSE, ...)

runsim.nonspatial (nrepl = 100, seed = NULL, ncores = 1, fitargs = list(),
    extractfn = predict, ...)

runsim.spatial (nrepl = 100, seed = NULL, ncores = 1, popargs = list(), detargs = list(),
    fitargs = list(), extractfn = predict)

sumsims (sims, parm = 'phi', session = 1, dropifnoSE = TRUE, svtol = NULL, maxcode = 3)

runsim.RMark (nrepl = 100, model = "CJS", model.parameters = NULL, extractfn,
    seed = NULL, ...)
```

Arguments

`N`	integer population size
`turnover`	list as described for turnover
`p`	numeric detection probability
`nsessions`	number of primary sessions
`noccasions`	number of secondary sessions per primary session
`intervals`	intervals between secondary sessions (see Details)
`recapfactor`	numeric multiplier for capture probability after first capture
`seed`	random number seed see random numbers
`savepopn`	logical; if TRUE the generated population is saved as an attribute of the capthist object
`...`	other arguments passed to `sim.popn` (sim.nonspatial) or `sim.nonspatial` (runsims)
`nrepl`	number of replicates
`ncores`	integer number of cores to be used for parallel processing
`popargs`	list of arguments for sim.popn
`detargs`	list of arguments for sim.capthist
`fitargs`	list of arguments for openCR.fit
`extractfn`	function applied to each fitted openCR model
`sims`	list output from `runsim.nonspatial` or `runsim.spatial`
`parm`	character name of parameter to summarise
`session`	integer vector of session numbers to summarise
`dropifnoSE`	logical; if TRUE then replicates are omitted when SE missing for parm
`svtol`	numeric; minimum singular value (eigenvalue) considered non-zero
`maxcode`	integer; maximum accepted value of convergence code
`model`	character; RMark model type
`model.parameters`	list with RMark model specification (see `?mark`)

Details

For `sim.nonspatial` – If `intervals` is specified then the number of primary and secondary sessions is inferred from `intervals` and `nsessions` and `noccasions` are ignored. If `N` and `p` are vectors of length 2 then subpopulations of the given initial size are sampled with the differing capture probabilities and the resulting capture histories are combined.

`runsim.spatial` is a relatively simple wrapper for `sim.popn`, `sim.capthist`, and `openCR.fit`. Some arguments are set automatically: the `sim.capthist` argument 'renumber' is always FALSE; argument 'seed' is ignored within 'popargs' and 'detargs'; if no 'traps' argument is provided in 'detargs' then 'core' from 'popargs' will be used; detargs$popn and fitargs$capthist are derived from the preceding step. The 'type' specified in fitargs may refer to a non-spatial or spatial open-population model ('CJS', 'JSSAsecrfCL' etc.).

Both `runsim.nonspatial` and `runsim.spatial` split replicates across multiple cores when 'ncores' > 1; 'ncores' is set to 1 for each replicate.

`sumsims` assumes output from `runsim.nonspatial` and `runsim.spatial` with 'extractfn = predict'. Missing SE usually reflects non-identifiability of a parameter or failure of maximisation, so these replicates are dropped by default. If `svtol` is specified then the rank of the Hessian is determined by counting eigenvalues that exceed svtol, and replicates are dropped if the rank is less than the number of beta parameters. A value of 1e-5 is suggested for svtol in `AIC.openCR`, but smaller values may be appropriate for larger models (MARK has its own algorithm for this threshold).

Replicates are also dropped by `sumsims` if the convergence code exceeds 'maxcode'. The maximisation functions `nlm` (used for method = 'Newton-Raphson', the default), and `optim` (all other methods) return different convergence codes; their help pages should be consulted. The default is to accept code = 3 from `nlm`, although the status of such maximisations is ambiguous.

Value

`sim.nonspatial` –

A capthist object representing an open-population sample

`runsim.nonspatial` and `runsim.spatial` –

List with one component (output from extractfn) for each replicate. Each component also has attributes 'eigH' and 'fit' taken from the output of `openCR.fit`. See Examples to extract convergence codes from 'fit' attribute.

See Also

`sim.popn`, `sim.capthist`

Examples

```
ch <- sim.nonspatial(100, list(phi = 0.7, lambda = 1.1), p = 0.3, nsessions = 8, noccasions=2)

openCR.fit(ch, type = 'CJS')


## Not run:

turnover <- list(phi = 0.85, lambda = 1.0, recrmodel = 'constantN')

## using type = 'JSSAlCL' and extractfn = predict

fitarg <- list(type = 'JSSAlCL', model = list(p~t, phi~t, lambda~t))
out <- runsim.nonspatial(nrepl = 100, N = 100, ncores = 6, turnover = turnover,
   p = 0.2, recapfactor = 4, nsessions = 10, noccasions = 1, fitargs = fitarg)
sumsims(out, 'lambda', 1:10)

## using type = 'Pradelg' and extractfn = derived
## homogeneous p
fitarg <- list(type = 'Pradelg', model = list(p~t, phi~t, gamma~t))
outg <- runsim.nonspatial(nrepl = 100, N = 100, ncores = 6, turnover = turnover,
    p = 0.2, recapfactor = 4, nsessions = 10, noccasions = 1,
    fitargs = fitarg, extractfn = derived)
apply(sapply(outg, function(x) x$estimates$lambda),1,mean)

turnover <- list(phi = 0.85, lambda = 1.0, recrmodel = 'discrete')

## 2-class mixture for p
outg2 <- runsim.nonspatial(nrepl = 100, N = c(50,50), ncores = 6, turnover = turnover,
    p = c(0.3,0.9), recapfactor = 1, nsessions = 10, noccasions = 1,
```

```
    fitargs = fitarg, extractfn = derived)
outg3 <- runsim.nonspatial(nrepl = 100, N = c(50,50), ncores = 6, turnover = turnover,
    p = c(0.3,0.3), recapfactor = 1, nsessions = 10, noccasions = 1,
    fitargs = fitarg, extractfn = derived)
apply(sapply(outg2, function(x) x$estimates$lambda),1,mean)

plot(2:10, apply(sapply(outg2, function(x) x$estimates$lambda),1,mean)[-1],
    type='o', xlim = c(1,10), ylim = c(0.9,1.1))

## RMark

extfn <- function(x) x$results$real$estimate[3:11]
MarkPath <- 'c:/mark'  ## customise as needed
turnover <- list(phi = 0.85, lambda = 1.0, recrmodel = 'discrete')
outrm <- runsim.RMark (nrepl = 100, model = 'Pradlambda', extractfn = extfn,
                       model.parameters = list(Lambda=list(formula=~time)),
                       N = c(200,200), turnover = turnover, p = c(0.3,0.9),
                       recapfactor = 1, nsessions = 10, noccasions = 1)
apply(do.call(rbind, outrm),1,mean)

## Spatial

grid <- make.grid()
msk <- make.mask(grid, type = 'trapbuffer', nx = 32)
turnover <- list(phi = 0.8, lambda = 1)
poparg <- list(D = 10, core = grid, buffer = 100, Ndist = 'fixed', nsessions = 6,
    details = turnover)
detarg <- list(noccasions = 5, detectfn = 'HHN', detectpar = list(lambda0 = 0.5, sigma = 20))
fitarg <- list(type = 'JSSAsecrfCL', mask = msk, model = list(phi~1, f~1))
sims <- runsim.spatial (nrepl = 7, ncores = 7, pop = poparg, det = detarg, fit = fitarg)
sumsims(sims)

## extract the convergence code from nlm for each replicate in preceding simulation
sapply(lapply(sims, attr, 'fit'), '[[', 'code')
## if method != 'Newton-Raphson then optim is used and the code is named 'convergence'
# sapply(lapply(sims, attr, 'fit'), '[[', 'convergence')


## End(Not run)
```

---

squeeze *Unique Capture Histories*

Description

Compresses or expands capthist objects.

Usage

```
squeeze(x)
unsqueeze(x)
```

Arguments

`x`	secr capthist object

Details

Although `squeeze` may be applied to spatial capthist objects, the effect is often minimal as most spatial histories are unique.

The 'freq' covariate is used by `openCR.fit` to weight summaries and likelihoods. It is currently ignored by `secr.fit`.

Value

Both functions return a capthist object with one row for each unique capture history (including covariates). The individual covariate 'freq' records the number of instances of each unique history in the input.

See Also

`openCR.fit`

Examples

```
squeeze(captdata)
```

Index