Package 'multimark'

July 23, 2025

Type Package

Title Capture-Mark-Recapture Analysis using Multiple Non-Invasive Marks

Version 2.1.6

Date 2023-03-09

Depends R (>= 3.2.1)

Imports parallel, Matrix, coda, statmod, RMark, Brobdingnag, mvtnorm, graphics, methods, stats, utils, prodlim, sp, raster

Description Traditional and spatial capture-mark-recapture analysis with multiple non-invasive marks. The models implemented in 'multimark' combine encounter history data arising from two different non-invasive ``marks'', such as images of left-sided and right-sided pelage patterns of bilaterally asymmetrical species, to estimate abundance and related demographic parameters while accounting for imperfect detection. Bayesian models are specified using simple formulae and fitted using Markov chain Monte Carlo. Addressing deficiencies in currently available software, 'multimark' also provides a user-friendly interface for performing Bayesian multimodel inference using non-spatial or spatial capture-recapture data consisting of a single conventional mark or multiple non-invasive marks. See McClintock (2015) <doi:10.1002/ece3.1676> and Maronde et al. (2020) <doi:10.1002/ece3.6990>.

Suggests testthat

License GPL-2

LazyData yes

ByteCompile TRUE

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author Brett T. McClintock [aut, cre],
Acho Arnold [ctb, cph] (C original matrix library, https://github.com/najela/matrix.h),
Barry Brown [ctb] (Fortran original ranlib library),
James Lovato [ctb] (Fortran original ranlib library),

John Burkardt [ctb] (C original ranlib library,
http://people.sc.fsu.edu/~jburkardt/c_src/ranlib),
Cleve Moler [ctb] (C original linpack library,
http://www.kkant.net/geist/ranlib/),
Arjun Gopalaswamy [ctb] (modified snippets of R package SPACECAP code)

Maintainer Brett T. McClintock <brett.mcclintock@noaa.gov>

Repository CRAN

Date/Publication 2023-03-10 11:20:06 UTC

Contents

bobcat . . . 2
bobcatSCR . . . 3
getdensityClosedSCR . . . 5
getprobsCJS . . . 6
getprobsClosed . . . 7
getprobsClosedSCR . . . 8
markCJS . . . 9
markClosed . . . 13
markClosedSCR . . . 16
multimarkCJS . . . 20
multimarkClosed . . . 26
multimarkClosedSCR . . . 31
multimarkSCRsetup-class . . . 37
multimarksetup-class . . . 39
multimodelCJS . . . 40
multimodelClosed . . . 43
multimodelClosedSCR . . . 45
plotSpatialData . . . 48
processdata . . . 49
processdataSCR . . . 51
simdataCJS . . . 54
simdataClosed . . . 56
simdataClosedSCR . . . 58
tiger . . . 62

Index **64**

`bobcat` *Bobcat data*

Description

Example bobcat data for `multimark` package.

Format

The data are summarized in a 46x8 matrix containing observed encounter histories for 46 bobcats across 8 sampling occasions. Bobcats are bilaterially asymmetrical, and sampling was conducted using camera stations consisting of a single camera.

Because the left-side cannot be reconciled with the right-side, the two types of "marks" in this case are the pelage patterns on the left- and right-side of each individual. Encounter type 0 corresponds to non-detection, encounter type 1 corresponds to left-sided detection, encounter type 2 corresponds to right-sided detection.

Both-sided encounters were never observed in this dataset, hence the most appropriate `multimark` data type is `data.type="never"`.

Source

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

See Also

`multimarkClosed`, `processdata`

Examples

```
data(bobcat)
```

`bobcatSCR` *Bobcat spatial capture-recapture data*

Description

Example spatial bobcat data for `multimark` package.

Format

These spatial capture-recapture data with multiple mark types are summarized in a list of length 3 containing the following objects:

`Enc.Mat` is a 42 x (noccas*ntraps) matrix containing observed encounter histories for 42 bobcats across `noccas=187` sampling occasions and `ntraps=30` traps. The first 187 columns correspond to trap 1, the second 187 columns coresopond to trap 2, etc.

`trapCoords` is a matrix of dimension `ntraps` x (2 + `noccas`) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate, and the second column the y-coordinate. The last `noccas` columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating.

`studyArea` is a 3-column matrix containing the coordinates for the centroids of the contiguous grid of 1023 cells that define the study area and available habitat. Each row corresponds to a grid cell.

The first 2 columns indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column indicates whether the cell is available habitat (=1) or not (=0). The grid cells are 0.65x0.65km resolution.

Bobcats are bilaterially asymmetrical, and sampling was conducted using camera stations consisting of a single camera. Because the left-side cannot be reconciled with the right-side, the two types of "marks" in this case are the pelage patterns on the left- and right-side of each individual. Encounter type 0 corresponds to non-detection, encounter type 1 corresponds to left-sided detection, encounter type 2 corresponds to right-sided detection.

Both-sided encounters were never observed in this dataset, hence the most appropriate `multimark` data type is `data.type="never"`.

The first 15 rows of `bobcatSCR$Enc.Mat` correspond to individuals for which both the left and right sides were known because they were physically captured for telemetry deployments prior to sampling surveys. The encounter histories for these 15 individuals are therefore known with certainty and should be specified as such using the `known` argument in `processdataSCR` and/or `multimarkClosedSCR` (see example below).

These data were obtained from the R package `SPIM` (Augustine et al. 2017) and modified by projecting onto a regular rectangular grid consisting of square grid cells (as is required by the spatial capture-recapture models in `multimark`).

Details

We thank B. Augustine and co-authors for making these data publicly available in the `SPIM` package (Augustine et al. 2017).

Source

Augustine, B., Royle, J.A., Kelly, M., Satter, C., Alonso, R., Boydston, E. and Crooks, K. 2017. Spatial capture-recapture with partial identity: an application to camera traps. bioRxiv doi: https://doi.org/10.1101/056804

See Also

`multimarkClosedSCR`, `processdataSCR`

Examples

```
data(bobcatSCR)
#plot the traps and available habitat within the study area
plotSpatialData(trapCoords=bobcatSCR$trapCoords,studyArea=bobcatSCR$studyArea)

# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

# Fit spatial model to tiger data
Enc.Mat <- bobcatSCR$Enc.Mat
trapCoords <- bobcatSCR$trapCoords
studyArea <- bobcatSCR$studyArea

# specify known encounter histories
known <- c(rep(1,15),rep(0,nrow(Enc.Mat)-15))
```

```
# specify prior bounds for sigma2_scr
sig_bounds <- c(0.1,max(diff(range(studyArea[,"x"])),diff(range(studyArea[,"y"]))))

mmsSCR <- processdataSCR(Enc.Mat,trapCoords,studyArea,known=known)
bobcatSCR.dot.type <- multimarkClosedSCR(mms=mmsSCR,iter=200,adapt=100,burnin=100,
                                         sigma_bounds=sig_bounds)
summary(bobcatSCR.dot.type$mcmc)
```

`getdensityClosedSCR` *Calculate population density estimates*

Description

This function calculates posterior population density estimates from `multimarkClosedSCR` output as D = N/A, where D is density, N is abundance, and A is the area of available habitat within the study area.

Usage

```
getdensityClosedSCR(out)
```

Arguments

`out`	List of output returned by `multimarkClosedSCR`.

Value

An object of class `mcmc.list` containing the following:

`D`	Posterior samples for density.

Author(s)

Brett T. McClintock

See Also

`multimarkClosedSCR`

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run behavior model for simulated data with constant detection probability (i.e., mod.p=~c)
sim.data<-simdataClosedSCR()
Enc.Mat<-sim.data$Enc.Mat
trapCoords<-sim.data$spatialInputs$trapCoords
studyArea<-sim.data$spatialInputs$studyArea
```

```
example.dot <- multimarkClosedSCR(Enc.Mat,trapCoords,studyArea,mod.p=~1)

#Calculate capture and recapture probabilities
D <- getdensityClosedSCR(example.dot)
summary(D)
```

---

getprobsCJS *Calculate posterior capture and survival probabilities*

Description

This function calculates posterior capture (p) and survival (ϕ) probabilities for each sampling occasion from multimarkCJS output.

Usage

```
getprobsCJS(out, link = "probit")
```

Arguments

out	List of output returned by multimarkCJS
link	Link function for p and ϕ. Must be "probit" or "logit". Note that multimarkCJS is currently implemented for the probit link only.

Value

An object of class mcmc.list containing the following:

p	Posterior samples for capture probability ($p[c,t]$) for each release cohort ($c = 1,\ldots,T-1$) and sampling occasion ($t = 2,\ldots,T$).
phi	Posterior samples for survival probability ($\phi[c,k]$) for each release cohort ($c = 1,\ldots,T-1$) and interval ($k = 1,\ldots,T-1$).

Author(s)

Brett T. McClintock

See Also

multimarkCJS

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Simulate open population data with temporal variation in survival
noccas <- 5
data <- simdataCJS(noccas=noccas, phibeta=rnorm(noccas-1,1.6,0.1))

#Fit open population model with temporal variation in survival
sim.time <- multimarkCJS(data$Enc.Mat,mod.phi=~time)

#Calculate capture and survival probabilities for each cohort and time
pphi <- getprobsCJS(sim.time)
summary(pphi)
```

`getprobsClosed` *Calculate posterior capture and recapture probabilities*

Description

This function calculates posterior capture (p) and recapture (c) probabilities for each sampling occasion from `multimarkClosed` output.

Usage

```
getprobsClosed(out, link = "logit")
```

Arguments

`out`	List of output returned by `multimarkClosed`.
`link`	Link function for detection probability. Must be "`logit`" or "`probit`". Note that `multimarkClosed` is currently implemented for the logit link only.

Value

An object of class `mcmc.list` containing the following:

`p`	Posterior samples for capture probability (p) for each sampling occasion.
`c`	Posterior samples for recapture probability (c) for each sampling occasion.

Author(s)

Brett T. McClintock

See Also

`multimarkClosed`

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run behavior model for bobcat data with constant detection probability (i.e., mod.p=~c)
bobcat.c <- multimarkClosed(bobcat,mod.p=~c)

#Calculate capture and recapture probabilities
pc <- getprobsClosed(bobcat.c)
summary(pc)
```

`getprobsClosedSCR` *Calculate posterior capture and recapture probabilities*

Description

This function calculates posterior spatial capture (p) and recapture (c) probabilities (at zero distance from an activity center) for each sampling occasion from `multimarkClosedSCR` output.

Usage

```
getprobsClosedSCR(out, link = "cloglog")
```

Arguments

`out`	List of output returned by `multimarkClosedSCR`.
`link`	Link function for detection probability. Must be "`cloglog`". Note that `multimarkClosedSCR` is currently implemented for the cloglog link only.

Value

An object of class `mcmc.list` containing the following:

`p`	Posterior samples for capture probability (p) for each sampling occasion (first index) and trap (second index).
`c`	Posterior samples for recapture probability (c) for each sampling occasion (first index) and trap (second index).

Author(s)

Brett T. McClintock

See Also

`multimarkClosedSCR`

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run behavior model for simulated data with constant detection probability (i.e., mod.p=~c)
sim.data<-simdataClosedSCR()
Enc.Mat<-sim.data$Enc.Mat
trapCoords<-sim.data$spatialInputs$trapCoords
studyArea<-sim.data$spatialInputs$studyArea
example.c <- multimarkClosedSCR(Enc.Mat,trapCoords,studyArea,mod.p=~c,
                                iter=1000,adapt=500,burnin=500)

#Calculate capture and recapture probabilities
pc <- getprobsClosedSCR(example.c)
summary(pc)
```

---

markCJS *Fit open population survival models for "traditional" capture-mark-recapture data consisting of a single mark type*

Description

This function fits Cormack-Jolly-Seber (CJS) open population models for survival probability (ϕ) and capture probability (p) for "traditional" capture-mark-recapture data consisting of a single mark type. Using Bayesian analysis methods, Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
markCJS(
  Enc.Mat,
  covs = data.frame(),
  mod.p = ~1,
  mod.phi = ~1,
  parms = c("pbeta", "phibeta"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propzp = 1,
  propsigmap = 1,
  propphibeta = 0.1,
```

```
  propzphi = 1,
  propsigmaphi = 1,
  pbeta0 = 0,
  pSigma0 = 1,
  phibeta0 = 0,
  phiSigma0 = 1,
  l0p = 1,
  d0p = 0.01,
  l0phi = 1,
  d0phi = 0.01,
  initial.values = NULL,
  link = "probit",
  printlog = FALSE,
  ...
)
```

Arguments

Enc.Mat	A matrix of observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions. With a single mark type, encounter histories consist of only non-detections (0) and type 1 encounters (1).
covs	A data frame of temporal covariates for detection probabilities (ignored unless mms=NULL). The number of rows in the data frame must equal the number of sampling occasions. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying mod.p and mod.phi.
mod.p	Model formula for detection probability (p). For example, mod.p=~1 specifies no effects (i.e., intercept only), mod.p~time specifies temporal effects, mod.p~age specifies age effects, mod.p~h specifies individual heterogeneity, and mod.p~time+age specifies additive temporal and age effects.
mod.phi	Model formula for survival probability (ϕ). For example, mod.phi=~1 specifies no effects (i.e., intercept only), mod.phi~time specifies temporal effects, mod.phi~age specifies age effects, mod.phi~h specifies individual heterogeneity, and mod.phi~time+age specifies additive temporal and age effects.
parms	A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are probit-scale detection probability parameters ("pbeta" for p and "phibeta" for ϕ), probit-scale individual heterogeneity variance terms ("sigma2_zp" for p and "sigma2_zphi" for ϕ), and probit-scale individual effects ("zp" and "zphi"). Latent variable indicators for whether each individual was alive (1) or dead (0) during each sampling occasion ("q") and the log likelihood ("loglike") may also be monitored. Setting parms="all" monitors all possible parameters and latent variables.
nchains	The number of parallel MCMC chains for the model.
iter	The number of MCMC iterations.
adapt	Ignored; no adaptive phase is needed for "probit" link.
bin	Ignored; no adaptive phase is needed for "probit" link.

thin	Thinning interval for monitored parameters.
burnin	Number of burn-in iterations (0 <= burnin < iter).
taccept	Ignored; no adaptive phase is needed for "probit" link.
tuneadjust	Ignored; no adaptive phase is needed for "probit" link.
proppbeta	Ignored; no adaptive phase is needed for "probit" link.
propzp	Ignored; no adaptive phase is needed for "probit" link.
propsigmap	Ignored; no adaptive phase is needed for "probit" link.
propphibeta	Ignored; no adaptive phase is needed for "probit" link.
propzphi	Ignored; no adaptive phase is needed for "probit" link.
propsigmaphi	Ignored; no adaptive phase is needed for "probit" link.
pbeta0	Scaler or vector (of length k) specifying mean of pbeta ~ multivariateNormal(pbeta0, pSigma0) prior. If pbeta0 is a scaler, then this value is used for all j = 1, ..., k. Default is pbeta0 = 0.
pSigma0	Scaler or k x k matrix specifying covariance matrix of pbeta ~ multivariateNormal(pbeta0, pSigma0) prior. If pSigma0 is a scaler, then this value is used for all pSigma0[j,j] for j = 1, ..., k (with pSigma[j,l] = 0 for all $j \neq l$). Default is pSigma0 = 1.
phibeta0	Scaler or vector (of length k) specifying mean of phibeta ~ multivariateNormal(phibeta0, phiSigma0) prior. If phibeta0 is a scaler, then this value is used for all j = 1, ..., k. Default is phibeta0 = 0.
phiSigma0	Scaler or k x k matrix specifying covariance matrix of phibeta ~ multivariateNormal(phibeta0, phiSigma0) prior. If phiSigma0 is a scaler, then this value is used for all phiSigma0[j,j] for j = 1, ..., k (with phiSigma[j,l] = 0 for all $j \neq l$). Default is phiSigma0 = 1.
l0p	Specifies "shape" parameter for [sigma2_zp] ~ invGamma(l0p,d0p) prior. Default is l0p = 1.
d0p	Specifies "scale" parameter for [sigma2_zp] ~ invGamma(l0p,d0p) prior. Default is d0p = 0.01.
l0phi	Specifies "shape" parameter for [sigma2_zphi] ~ invGamma(l0phi,d0phi) prior. Default is l0phi = 1.
d0phi	Specifies "scale" parameter for [sigma2_zphi] ~ invGamma(l0phi,d0phi) prior. Default is d0phi = 0.01.
initial.values	OOptional list of nchain list(s) specifying initial values for "pbeta", "phibeta", "sigma2_zp", "sigma2_zphi", "zp", "zphi", and "q". Default is initial.values = NULL, which causes initial values to be generated automatically.
link	Link function for survival and capture probabilities. Only probit link is currently implemented.
printlog	Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.
...	Additional "parameters" arguments for specifying mod.p and mod.phi. See RMark::make.design.data.

Details

The first time markCJS (or markClosed) is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in multimarkCJS. Note that setting parms="all" is required for any markCJS model output to be used in multimodelCJS.

Value

A list containing the following:

mcmc	Markov chain Monte Carlo object of class mcmc.list.
mod.p	Model formula for detection probability (as specified by mod.p above).
mod.phi	Model formula for survival probability (as specified by mod.phi above).
mod.delta	Formula always NULL; only for internal use in multimodelCJS.
DM	A list of design matrices for detection and survival probability respectively generated by mod.p and mod.phi, where DM$p is the design matrix for capture probability (p) and DM$phi is the design matrix for survival probability (ϕ).
initial.values	A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "phibeta", "sigma2_zp", "sigma2_zphi", "zp", "zphi", and "q".
mms	An object of class multimarksetup

Author(s)

Brett T. McClintock

See Also

processdata, multimodelCJS

Examples

```
# These examples are excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Simulate open population data using defaults
data <- simdataCJS(delta_1=1,delta_2=0)$Enc.Mat

#Fit default open population model
sim.dot <- markCJS(data)

#Posterior summary for monitored parameters
summary(sim.dot$mcmc)
plot(sim.dot$mcmc)

#Fit ``age'' model with 2 age classes (e.g., juvenile and adult) for survival
#using 'parameters' and 'right' arguments from RMark::make.design.data
sim.age <- markCJS(data,mod.phi=~age,
           parameters=list(Phi=list(age.bins=c(0,1,4))),right=FALSE)
```

```
summary(getprobsCJS(sim.age))
```

`markClosed` *Fit closed population abundance models for "traditional" capture-mark-recapture data consisting of a single mark type*

Description

This function fits closed population abundance models for "traditional" capture-mark-recapture data consisting of a single mark type using Bayesian analysis methods. Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
markClosed(
  Enc.Mat,
  covs = data.frame(),
  mod.p = ~1,
  parms = c("pbeta", "N"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propzp = 1,
  propsigmap = 1,
  npoints = 500,
  a = 25,
  mu0 = 0,
  sigma2_mu0 = 1.75,
  initial.values = NULL,
  printlog = FALSE,
  ...
)
```

Arguments

`Enc.Mat`	A matrix of observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions. With a single mark type, encounter histories consist of only non-detections (0) and type 1 encounters (1).

covs	A data frame of temporal covariates for detection probabilities (ignored unless mms=NULL). The number of rows in the data frame must equal the number of sampling occasions. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying mod.p and mod.phi.
mod.p	Model formula for detection probability. For example, mod.p=~1 specifies no effects (i.e., intercept only), mod.p~time specifies temporal effects, mod.p~c specifies behavioral reponse (i.e., trap "happy" or "shy"), mod.p~h specifies individual heterogeneity, and mod.p~time+c specifies additive temporal and behavioral effects.
parms	A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are logit-scale detection probability parameters ("pbeta"), population abundance ("N"), logit-scale individual heterogeneity variance term ("sigma2_zp"), and logit-scale individual effects ("zp"). The log posterior density ("logPosterior") may also be monitored. Setting parms="all" monitors all possible parameters and latent variables.
nchains	The number of parallel MCMC chains for the model.
iter	The number of MCMC iterations.
adapt	The number of iterations for proposal distribution adaptation. If adapt = 0 then no adaptation occurs.
bin	Bin length for calculating acceptance rates during adaptive phase (0 < bin <= iter).
thin	Thinning interval for monitored parameters.
burnin	Number of burn-in iterations (0 <= burnin < iter).
taccept	Target acceptance rate during adaptive phase (0 < taccept <= 1). Acceptance rate is monitored every bin iterations. Default is taccept = 0.44.
tuneadjust	Adjustment term during adaptive phase (0 < tuneadjust <= 1). If acceptance rate is less than taccept, then proposal term (proppbeta, propzp, or propsigmap) is multiplied by tuneadjust. If acceptance rate is greater than or equal to taccept, then proposal term is divided by tuneadjust. Default is tuneadjust = 0.95.
proppbeta	Scaler or vector (of length k) specifying the initial standard deviation of the Normal(pbeta[j], proppbeta[j]) proposal distribution. If proppbeta is a scaler, then this value is used for all j = 1, ..., k. Default is proppbeta = 0.1.
propzp	Scaler or vector (of length M) specifying the initial standard deviation of the Normal(zp[i], propzp[i]) proposal distribution. If propzp is a scaler, then this value is used for all i = 1, ..., M individuals. Default is propzp = 1.
propsigmap	Scaler specifying the initial Gamma(shape = 1/propsigmap, scale = sigma_zp * propsigmap) proposal distribution for sigma_zp = sqrt(sigma2_zp). Default is propsigmap=1.
npoints	Number of Gauss-Hermite quadrature points to use for numerical integration. Accuracy increases with number of points, but so does computation time.
a	Scale parameter for [sigma_z] ~ half-Cauchy(a) prior for the individual hetegeneity term sigma_zp = sqrt(sigma2_zp). Default is “uninformative” a = 25.

mu0	Scaler or vector (of length k) specifying mean of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If mu0 is a scaler, then this value is used for all j = 1, ..., k. Default is mu0 = 0.
sigma2_mu0	Scaler or vector (of length k) specifying variance of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If sigma2_mu0 is a scaler, then this value is used for all j = 1, ..., k. Default is sigma2_mu0 = 1.75.
initial.values	Optional list of nchain list(s) specifying initial values for "pbeta", "zp", "sigma2_zp", and "N". Default is initial.values = NULL, which causes initial values to be generated automatically.
printlog	Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.
...	Additional "parameters" arguments for specifying mod.p. See make.design.data.

Details

The first time markClosed (or markCJS) is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in markClosed. Note that setting parms="all" is required for any markClosed model output to be used in multimodelClosed.

Value

A list containing the following:

mcmc	Markov chain Monte Carlo object of class mcmc.list.
mod.p	Model formula for detection probability (as specified by mod.p above).
mod.delta	Formula always NULL; only for internal use in multimodelClosed.
DM	A list of design matrices for detection probability generated for model mod.p, where DM$p is the design matrix for initial capture probability (p) and DM$c is the design matrix for recapture probability (c).
initial.values	A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "zp", "sigma2_zp", and "N".
mms	An object of class multimarksetup

Author(s)

Brett T. McClintock

See Also

multimodelClosed

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run single chain using the default model for simulated ``traditional'' data
data<-simdataClosed(delta_1=1,delta_2=0)$Enc.Mat
sim.dot<-markClosed(data)

#Posterior summary for monitored parameters
summary(sim.dot$mcmc)
plot(sim.dot$mcmc)
```

`markClosedSCR` *Fit spatial population abundance models for "traditional" capture-mark-recapture data consisting of a single mark type*

Description

This function fits spatial population abundance models for "traditional" capture-mark-recapture data consisting of a single mark type using Bayesian analysis methods. Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
markClosedSCR(
  Enc.Mat,
  trapCoords,
  studyArea = NULL,
  buffer = NULL,
  ncells = 1024,
  covs = data.frame(),
  mod.p = ~1,
  detection = "half-normal",
  parms = c("pbeta", "N"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propsigma = 1,
  propcenter = NULL,
  sigma_bounds = NULL,
```

```
  mu0 = 0,
  sigma2_mu0 = 1.75,
  initial.values = NULL,
  scalemax = 10,
  printlog = FALSE,
  ...
)
```

Arguments

Enc.Mat	A matrix containing the observed encounter histories with rows corresponding to individuals and (ntraps*noccas) columns corresponding to traps and sampling occasions. The first noccas columns correspond to trap 1, the second noccas columns coresopond to trap 2, etc.
trapCoords	A matrix of dimension ntraps x (2 + noccas) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate ("x"), and the second column the y-coordinate ("y"). The last noccas columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating. Ignored unless mms=NULL.
studyArea	is a 3-column matrix containing the coordinates for the centroids a contiguous grid of cells that define the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns ("x" and "y") indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column ("avail") indicates whether the cell is available habitat (=1) or not (=0). All cells must have the same resolution. If studyArea=NULL (the default) and mms=NULL, then a square study area grid composed of ncells cells of available habitat is drawn around the bounding box of trapCoords based on buffer. Ignored unless mms=NULL. Note that rows should be ordered by raster cell order (raster cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom).
buffer	A scaler in same units as trapCoords indicating the buffer around the bounding box of trapCoords for defining the study area when studyArea=NULL. Ignored unless studyArea=NULL.
ncells	The number of grid cells in the study area when studyArea=NULL. The square root of ncells must be a whole number. Default is ncells=1024. Ignored unless studyArea=NULL and mms=NULL.
covs	A data frame of time- and/or trap-dependent covariates for detection probabilities (ignored unless mms=NULL). The number of rows in the data frame must equal the number of traps times the number of sampling occasions (ntraps*noccas), where the first noccas rows correspond to trap 1, the noccas rows correspond to trap 2, etc. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying mod.p and mod.phi.
mod.p	Model formula for detection probability. For example, mod.p=~1 specifies no effects (i.e., intercept only), mod.p~time specifies temporal effects, mod.p~c

specifies behavioral reponse (i.e., trap "happy" or "shy"), `mod.p~trap` specifies trap effects, and `mod.p~time+c` specifies additive temporal and behavioral effects.

`detection` Model for detection probability as a function of distance from activity centers . Must be "`half-normal`" (of the form $\exp(-d^2/(2*\sigma^2))$, where d is distance) or "`exponential`" (of the form $\exp(-d/\lambda)$).

`parms` A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are cloglog-scale detection probability parameters ("`pbeta`"), population abundance ("`N`"), and cloglog-scale distance term for the detection function ("`sigma2_scr`" when `detection=``half-normal''` or "`lambda`" when `detection=``exponential''`). Individual activity centers ("`centers`") and the log posterior density ("`logPosterior`") may also be monitored. Setting `parms="all"` monitors all possible parameters and latent variables.

`nchains` The number of parallel MCMC chains for the model.

`iter` The number of MCMC iterations.

`adapt` The number of iterations for proposal distribution adaptation. If `adapt = 0` then no adaptation occurs.

`bin` Bin length for calculating acceptance rates during adaptive phase (`0 < bin <= iter`).

`thin` Thinning interval for monitored parameters.

`burnin` Number of burn-in iterations (`0 <= burnin < iter`).

`taccept` Target acceptance rate during adaptive phase (`0 < taccept <= 1`). Acceptance rate is monitored every `bin` iterations. Default is `taccept = 0.44`.

`tuneadjust` Adjustment term during adaptive phase (`0 < tuneadjust <= 1`). If acceptance rate is less than `taccept`, then proposal term (`proppbeta` or `propsigma`) is multiplied by `tuneadjust`. If acceptance rate is greater than or equal to `taccept`, then proposal term is divided by `tuneadjust`. Default is `tuneadjust = 0.95`.

`proppbeta` Scaler or vector (of length k) specifying the initial standard deviation of the Normal(pbeta[j], proppbeta[j]) proposal distribution. If `proppbeta` is a scaler, then this value is used for all j = 1, ..., k. Default is `proppbeta = 0.1`.

`propsigma` Scaler specifying the initial Gamma(shape = 1/`propsigma`, scale = sigma_scr * `propsigma`) proposal distribution for sigma_scr = sqrt(sigma2_scr). Default is `propsigma=1`.

`propcenter` Scaler specifying the neighborhood distance when proposing updates to activity centers. When `propcenter=NULL` (the default), then propcenter = a*10, where a is the cell size for the study area grid, and each cell has (at most) approximately 300 neighbors.

`sigma_bounds` Positive vector of length 2 for the lower and upper bounds for the [sigma_scr] ~ Uniform(sigma_bounds[1], sigma_bounds[2]) (or [sqrt(lambda)] when `detection=``exponential''`) prior for the detection function term sigma_scr = sqrt(sigma2_scr) (or sqrt(lambda)). When `sigma_bounds = NULL` (the default), then `sigma_bounds = c(1.e-6,max(diff(range(studyArea`

`mu0` Scaler or vector (of length k) specifying mean of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If `mu0` is a scaler, then this value is used for all j = 1, ..., k. Default is `mu0 = 0`.

sigma2_mu0	Scaler or vector (of length k) specifying variance of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If sigma2_mu0 is a scaler, then this value is used for all j = 1, ..., k. Default is sigma2_mu0 = 1.75.
initial.values	Optional list of nchain list(s) specifying initial values for "pbeta", "N", "sigma2_scr", and "centers". Default is initial.values = NULL, which causes initial values to be generated automatically.
scalemax	Upper bound for internal re-scaling of grid cell centroid coordinates. Default is scalemax=10, which re-scales the centroids to be between 0 and 10. Re-scaling is done internally to avoid numerical overflows during model fitting.
printlog	Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.
...	Additional "parameters" arguments for specifying mod.p. See make.design.data.

Details

The first time markClosedSCR is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in markClosed. Note that setting parms="all" is required for any markClosed model output to be used in multimodelClosed.

Value

A list containing the following:

mcmc	Markov chain Monte Carlo object of class mcmc.list.
mod.p	Model formula for detection probability (as specified by mod.p above).
mod.delta	Formula always NULL; only for internal use in multimodelClosedSCR.
mod.det	Model formula for detection function (as specified by detection above).
DM	A list of design matrices for detection probability generated for model mod.p, where DM$p is the design matrix for initial capture probability (p) and DM$c is the design matrix for recapture probability (c).
initial.values	A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "N", "sigma2_scr", and "centers".
mms	An object of class multimarkSCRsetup

Author(s)

Brett T. McClintock

References

Gopalaswamy, A.M., Royle, J.A., Hines, J.E., Singh, P., Jathanna, D., Kumar, N. and Karanth, K.U. 2012. Program SPACECAP: software for estimating animal density using spatially explicit capture-recapture models. *Methods in Ecology and Evolution* 3:1067-1072.

King, R., McClintock, B. T., Kidney, D., and Borchers, D. L. 2016. Capture-recapture abundance estimation using a semi-complete data likelihood approach. *The Annals of Applied Statistics* 10: 264-285

Royle, J.A., Karanth, K.U., Gopalaswamy, A.M. and Kumar, N.S. 2009. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90: 3233-3244.

See Also

multimodelClosedSCR

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run single chain using the default model for ``traditional'' tiger data of Royle et al (2009)
Enc.Mat<-tiger$Enc.Mat
trapCoords<-tiger$trapCoords
studyArea<-tiger$studyArea
tiger.dot<-markClosedSCR(Enc.Mat,trapCoords,studyArea,iter=100,adapt=50,burnin=50)

#Posterior summary for monitored parameters
summary(tiger.dot$mcmc)
plot(tiger.dot$mcmc)
```

---

multimarkCJS — *Fit open population survival models for capture-mark-recapture data consisting of multiple non-invasive marks*

Description

This function fits Cormack-Jolly-Seber (CJS) open population models for survival probability (ϕ) and capture probability (p) from capture-mark-recapture data consisting of multiple non-invasive marks. Using Bayesian analysis methods, Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
multimarkCJS(
  Enc.Mat,
  data.type = "never",
  covs = data.frame(),
```

```
  mms = NULL,
  mod.p = ~1,
  mod.phi = ~1,
  mod.delta = ~type,
  parms = c("pbeta", "phibeta", "delta"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propzp = 1,
  propsigmap = 1,
  propphibeta = 0.1,
  propzphi = 1,
  propsigmaphi = 1,
  maxnumbasis = 1,
  pbeta0 = 0,
  pSigma0 = 1,
  phibeta0 = 0,
  phiSigma0 = 1,
  l0p = 1,
  d0p = 0.01,
  l0phi = 1,
  d0phi = 0.01,
  a0delta = 1,
  a0alpha = 1,
  b0alpha = 1,
  a0psi = 1,
  b0psi = 1,
  initial.values = NULL,
  known = integer(),
  link = "probit",
  printlog = FALSE,
  ...
)
```

Arguments

`Enc.Mat`	A matrix of observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions (ignored unless `mms=NULL`).
`data.type`	Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3

encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted:

`data.type="never"` indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See `bobcat`. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3).

`data.type="sometimes"` indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

`data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`covs` A data frame of temporal covariates for detection probabilities (ignored unless `mms=NULL`). The number of rows in the data frame must equal the number of sampling occasions. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying `mod.p` and `mod.phi`.

`mms` An optional object of class `multimarksetup-class`; if `NULL` it is created. See `processdata`.

`mod.p` Model formula for detection probability (p). For example, `mod.p=~1` specifies no effects (i.e., intercept only), `mod.p~time` specifies temporal effects, `mod.p~age` specifies age effects, `mod.p~h` specifies individual heterogeneity, and `mod.p~time+age` specifies additive temporal and age effects.

`mod.phi` Model formula for survival probability (ϕ). For example, `mod.phi=~1` specifies no effects (i.e., intercept only), `mod.phi~time` specifies temporal effects, `mod.phi~age` specifies age effects, `mod.phi~h` specifies individual heterogeneity, and `mod.phi~time+age` specifies additive temporal and age effects.

`mod.delta` Model formula for conditional probabilities of type 1 (delta_1) and type 2 (delta_2) encounters, given detection. Currently only `mod.delta=~1` (i.e., $\delta_1 = \delta_2$) and `mod.delta=~type` (i.e., $\delta_1 \neq \delta_2$) are implemented.

`parms` A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are probit-scale detection probability parameters ("`pbeta`" for p and "`phibeta`" for ϕ), conditional probability of type 1 or type 2 encounter, given detection ("`delta`)", probability of simultaneous type 1 and type 2 detection, given both types encountered ("`alpha`"), probit-scale individual heterogeneity variance terms ("`sigma2_zp`" for p and "`sigma2_zphi`" for

	ϕ), probit-scale individual effects ("zp" and "zphi"), and the probability that a randomly selected individual from the M = nrow(Enc.Mat) observed individuals belongs to the n unique individuals encountered at least once ("psi"). Individual encounter history indices ("H"), latent variable indicators for whether each individual was alive (1) or dead (0) during each sampling occasion ("q"), and the log likelihood ("loglike") may also be monitored. Setting parms="all" monitors all possible parameters and latent variables.
nchains	The number of parallel MCMC chains for the model.
iter	The number of MCMC iterations.
adapt	Ignored; no adaptive phase is needed for "probit" link.
bin	Ignored; no adaptive phase is needed for "probit" link.
thin	Thinning interval for monitored parameters.
burnin	Number of burn-in iterations (0 <= burnin < iter).
taccept	Ignored; no adaptive phase is needed for "probit" link.
tuneadjust	Ignored; no adaptive phase is needed for "probit" link.
proppbeta	Ignored; no adaptive phase is needed for "probit" link.
propzp	Ignored; no adaptive phase is needed for "probit" link.
propsigmap	Ignored; no adaptive phase is needed for "probit" link.
propphibeta	Ignored; no adaptive phase is needed for "probit" link.
propzphi	Ignored; no adaptive phase is needed for "probit" link.
propsigmaphi	Ignored; no adaptive phase is needed for "probit" link.
maxnumbasis	Maximum number of basis vectors to use when proposing latent history frequency updates. Default is maxnumbasis = 1, but higher values can potentially improve mixing.
pbeta0	Scaler or vector (of length k) specifying mean of pbeta ~ multivariateNormal(pbeta0, pSigma0) prior. If pbeta0 is a scaler, then this value is used for all j = 1, ..., k. Default is pbeta0 = 0.
pSigma0	Scaler or k x k matrix specifying covariance matrix of pbeta ~ multivariateNormal(pbeta0, pSigma0) prior. If pSigma0 is a scaler, then this value is used for all pSigma0[j,j] for j = 1, ..., k (with pSigma[j,l] = 0 for all $j \neq l$). Default is pSigma0 = 1.
phibeta0	Scaler or vector (of length k) specifying mean of phibeta ~ multivariateNormal(phibeta0, phiSigma0) prior. If phibeta0 is a scaler, then this value is used for all j = 1, ..., k. Default is phibeta0 = 0.
phiSigma0	Scaler or k x k matrix specifying covariance matrix of phibeta ~ multivariateNormal(phibeta0, phiSigma0) prior. If phiSigma0 is a scaler, then this value is used for all phiSigma0[j,j] for j = 1, ..., k (with phiSigma[j,l] = 0 for all $j \neq l$). Default is phiSigma0 = 1.
l0p	Specifies "shape" parameter for [sigma2_zp] ~ invGamma(l0p,d0p) prior. Default is l0p = 1.
d0p	Specifies "scale" parameter for [sigma2_zp] ~ invGamma(l0p,d0p) prior. Default is d0p = 0.01.

`l0phi` Specifies "shape" parameter for [sigma2_zphi] ~ invGamma(l0phi,d0phi) prior. Default is `l0phi = 1`.

`d0phi` Specifies "scale" parameter for [sigma2_zphi] ~ invGamma(l0phi,d0phi) prior. Default is `d0phi = 0.01`.

`a0delta` Scaler or vector (of length d) specifying the prior for the conditional (on detection) probability of type 1 (delta_1), type 2 (delta_2), and both type 1 and type 2 encounters (1-delta_1-delta_2). If `a0delta` is a scaler, then this value is used for all a0delta[j] for j = 1, ..., d. For `mod.delta=~type`, d=3 with [delta_1, delta_2, 1-delta_1-delta_2] ~ Dirichlet(a0delta) prior. For `mod.delta=~1`, d=2 with [tau] ~ Beta(a0delta[1],a0delta[2]) prior, where (delta_1,delta_2,1-delta_1-delta_2) = (tau/2,tau/2,1-tau). See McClintock et al. (2013) for more details.

`a0alpha` Specifies "shape1" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `a0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`b0alpha` Specifies "shape2" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `b0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`a0psi` Specifies "shape1" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `a0psi = 1`.

`b0psi` Specifies "shape2" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `b0psi = 1`.

`initial.values` Optional list of `nchain` list(s) specifying initial values for parameters and latent variables. Default is `initial.values = NULL`, which causes initial values to be generated automatically. In addition to the parameters ("`pbeta`", "`phibeta`", "`delta_1`", "`delta_2`", "`alpha`", "`sigma2_zp`", "`sigma2_zphi`", "`zp`", "`zphi`", and "`psi`"), initial values can be specified for the initial latent history frequencies ("`x`"), initial individual encounter history indices ("`H`"), and initial latent variable indicators for whether each individual was alive (1) or dead (0) during each sampling occasion ("`q`").

`known` Optional integer vector indicating whether the encounter history of an individual is known with certainty (i.e., the observed encounter history is the true encounter history). Encounter histories with at least one type 4 encounter are automatically assumed to be known, and `known` does not need to be specified unless there exist encounter histories that do not contain a type 4 encounter that happen to be known with certainty (e.g., from independent telemetry studies). If specified, `known = c(v_1,v_2,...,v_M)` must be a vector of length `M = nrow(Enc.Mat)` where `v_i = 1` if the encounter history for individual `i` is known (`v_i = 0` otherwise). Note that known all-zero encounter histories (e.g., '000') are ignored.

`link` Link function for survival and capture probabilities. Only probit link is currently implemented.

`printlog` Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when `nchains=1`. Updates are printed to log file as 1% increments of `iter` of each chain are completed. With >1 chains, setting `printlog=TRUE` is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when `printlog=FALSE`. Default is `printlog=FALSE`.

...	Additional "parameters" arguments for specifying mod.p and mod.phi. See RMark::make.design.data.

Details

The first time multimarkCJS (or multimarkClosed) is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in multimarkCJS. Note that setting parms="all" is required for any multimarkCJS model output to be used in multimodelCJS.

Value

A list containing the following:

mcmc	Markov chain Monte Carlo object of class mcmc.list.
mod.p	Model formula for detection probability (as specified by mod.p above).
mod.phi	Model formula for survival probability (as specified by mod.phi above).
mod.delta	Formula always NULL; only for internal use in multimodelCJS.
DM	A list of design matrices for detection and survival probability respectively generated by mod.p and mod.phi, where DM$p is the design matrix for capture probability (p) and DM$phi is the design matrix for survival probability (ϕ).
initial.values	A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "phibeta", "delta_1", "delta_2", "alpha", "sigma2_zp" "sigma2_zphi", "zp", "zphi", "psi", "x", "H", and "q".
mms	An object of class multimarksetup

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

McClintock, B. T., Bailey, L. L., Dreher, B. P., and Link, W. A. 2014. Probit models for capture-recapture data subject to imperfect detection, individual heterogeneity and misidentification. *The Annals of Applied Statistics* 8: 2461-2484.

See Also

processdata, multimodelCJS

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Simulate open population data using defaults
data <- simdataCJS()

#Fit default open population model
sim.dot <- multimarkCJS(data$Enc.Mat)

#Posterior summary for monitored parameters
summary(sim.dot$mcmc)
plot(sim.dot$mcmc)

#' #Fit ``age'' model with 2 age classes (e.g., juvenile and adult) for survival
#using 'parameters' and 'right' arguments from RMark::make.design.data
sim.age <- multimarkCJS(data$Enc.Mat,mod.phi=~age,
           parameters=list(Phi=list(age.bins=c(0,1,4))),right=FALSE)
summary(getprobsCJS(sim.age))
```

`multimarkClosed` *Fit closed population abundance models for capture-mark-recapture data consisting of multiple non-invasive marks*

Description

This function fits closed population abundance models for capture-mark-recapture data consisting of multiple non-invasive marks using Bayesian analysis methods. Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
multimarkClosed(
  Enc.Mat,
  data.type = "never",
  covs = data.frame(),
  mms = NULL,
  mod.p = ~1,
  mod.delta = ~type,
  parms = c("pbeta", "delta", "N"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
```

```
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propzp = 1,
  propsigmap = 1,
  npoints = 500,
  maxnumbasis = 1,
  a0delta = 1,
  a0alpha = 1,
  b0alpha = 1,
  a = 25,
  mu0 = 0,
  sigma2_mu0 = 1.75,
  a0psi = 1,
  b0psi = 1,
  initial.values = NULL,
  known = integer(),
  printlog = FALSE,
  ...
)
```

Arguments

`Enc.Mat`	A matrix of observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions (ignored unless `mms=NULL`).
`data.type`	Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted: `data.type="never"` indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See `bobcat`. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3). `data.type="sometimes"` indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). `data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0),

type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`covs` A data frame of temporal covariates for detection probabilities (ignored unless `mms=NULL`). The number of rows in the data frame must equal the number of sampling occasions. Covariate names cannot be "time", "c", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying `mod.p` and `mod.phi`.

`mms` An optional object of class `multimarksetup-class`; if `NULL` it is created. See `processdata`.

`mod.p` Model formula for detection probability. For example, `mod.p=~1` specifies no effects (i.e., intercept only), `mod.p~time` specifies temporal effects, `mod.p~c` specifies behavioral reponse (i.e., trap "happy" or "shy"), `mod.p~h` specifies individual heterogeneity, and `mod.p~time+c` specifies additive temporal and behavioral effects.

`mod.delta` Model formula for conditional probabilities of type 1 (delta_1) and type 2 (delta_2) encounters, given detection. Currently only `mod.delta=~1` (i.e., $\delta_1 = \delta_2$) and `mod.delta=~type` (i.e., $\delta_1 \neq \delta_2$) are implemented.

`parms` A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are logit-scale detection probability parameters ("`pbeta`"), population abundance ("`N`"), conditional probability of type 1 or type 2 encounter, given detection ("`delta`)", probability of simultaneous type 1 and type 2 detection, given both types encountered ("`alpha`"), logit-scale individual heterogeneity variance term ("`sigma2_zp`"), logit-scale individual effects ("`zp`"), and the probability that a randomly selected individual from the `M = nrow(Enc.Mat)` observed individuals belongs to the n unique individuals encountered at least once ("`psi`"). Individual encounter history indices ("`H`") and the log posterior density ("`logPosterior`") may also be monitored. Setting `parms="all"` monitors all possible parameters and latent variables.

`nchains` The number of parallel MCMC chains for the model.

`iter` The number of MCMC iterations.

`adapt` The number of iterations for proposal distribution adaptation. If `adapt = 0` then no adaptation occurs.

`bin` Bin length for calculating acceptance rates during adaptive phase (`0 < bin <= iter`).

`thin` Thinning interval for monitored parameters.

`burnin` Number of burn-in iterations (`0 <= burnin < iter`).

`taccept` Target acceptance rate during adaptive phase (`0 < taccept <= 1`). Acceptance rate is monitored every `bin` iterations. Default is `taccept = 0.44`.

`tuneadjust` Adjustment term during adaptive phase (`0 < tuneadjust <= 1`). If acceptance rate is less than `taccept`, then proposal term (`proppbeta`, `propzp`, or `propsigmap`) is multiplied by `tuneadjust`. If acceptance rate is greater than or equal to `taccept`, then proposal term is divided by `tuneadjust`. Default is `tuneadjust = 0.95`.

`proppbeta` Scaler or vector (of length k) specifying the initial standard deviation of the Normal(pbeta[j], proppbeta[j]) proposal distribution. If `proppbeta` is a scaler, then this value is used for all j = 1, ..., k. Default is `proppbeta = 0.1`.

`propzp` Scaler or vector (of length M) specifying the initial standard deviation of the Normal(zp[i], propzp[i]) proposal distribution. If `propzp` is a scaler, then this value is used for all i = 1, ..., M individuals. Default is `propzp = 1`.

`propsigmap` Scaler specifying the initial Gamma(shape = 1/`propsigmap`, scale = sigma_zp * `propsigmap`) proposal distribution for sigma_zp = sqrt(sigma2_zp). Default is `propsigmap=1`.

`npoints` Number of Gauss-Hermite quadrature points to use for numerical integration. Accuracy increases with number of points, but so does computation time.

`maxnumbasis` Maximum number of basis vectors to use when proposing latent history frequency updates. Default is `maxnumbasis = 1`, but higher values can potentially improve mixing.

`a0delta` Scaler or vector (of length d) specifying the prior for the conditional (on detection) probability of type 1 (delta_1), type 2 (delta_2), and both type 1 and type 2 encounters (1-delta_1-delta_2). If `a0delta` is a scaler, then this value is used for all a0delta[j] for j = 1, ..., d. For `mod.delta=~type`, d=3 with [delta_1, delta_2, 1-delta_1-delta_2] ~ Dirichlet(a0delta) prior. For `mod.delta=~1`, d=2 with [tau] ~ Beta(a0delta[1],a0delta[2]) prior, where (delta_1,delta_2,1-delta_1-delta_2) = (tau/2,tau/2,1-tau). See McClintock et al. (2013) for more details.

`a0alpha` Specifies "shape1" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `a0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`b0alpha` Specifies "shape2" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `b0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`a` Scale parameter for [sigma_z] ~ half-Cauchy(a) prior for the individual hetegeneity term sigma_zp = sqrt(sigma2_zp). Default is "uninformative" `a = 25`.

`mu0` Scaler or vector (of length k) specifying mean of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If `mu0` is a scaler, then this value is used for all j = 1, ..., k. Default is `mu0 = 0`.

`sigma2_mu0` Scaler or vector (of length k) specifying variance of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If `sigma2_mu0` is a scaler, then this value is used for all j = 1, ..., k. Default is `sigma2_mu0 = 1.75`.

`a0psi` Specifies "shape1" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `a0psi = 1`.

`b0psi` Specifies "shape2" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `b0psi = 1`.

`initial.values` Optional list of `nchain` list(s) specifying initial values for parameters and latent variables. Default is `initial.values = NULL`, which causes initial values to be generated automatically. In addition to the parameters ("`pbeta`", "`N`", "`delta_1`", "`delta_2`", "`alpha`", "`sigma2_zp`", "`zp`", and "`psi`"), initial values can be specified for the initial latent history frequencies ("`x`") and initial individual encounter history indices ("`H`").

known — Optional integer vector indicating whether the encounter history of an individual is known with certainty (i.e., the observed encounter history is the true encounter history). Encounter histories with at least one type 4 encounter are automatically assumed to be known, and known does not need to be specified unless there exist encounter histories that do not contain a type 4 encounter that happen to be known with certainty (e.g., from independent telemetry studies). If specified, known = c(v_1,v_2,...,v_M) must be a vector of length M = nrow(Enc.Mat) where v_i = 1 if the encounter history for individual i is known (v_i = 0 otherwise). Note that known all-zero encounter histories (e.g., '000') are ignored.

printlog — Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.

... — Additional "parameters" arguments for specifying mod.p. See make.design.data.

Details

The first time multimarkClosed (or multimarkCJS) is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in multimarkClosed. Note that setting parms="all" is required for any multimarkClosed model output to be used in multimodelClosed.

Value

A list containing the following:

mcmc — Markov chain Monte Carlo object of class mcmc.list.

mod.p — Model formula for detection probability (as specified by mod.p above).

mod.delta — Model formula for conditional probability of type 1 or type 2 encounter, given detection (as specified by mod.delta above).

DM — A list of design matrices for detection probability generated for model mod.p, where DM$p is the design matrix for initial capture probability (p) and DM$c is the design matrix for recapture probability (c).

initial.values — A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "N", "delta_1", "delta_2", "alpha", "sigma2_zp", "zp", "psi", "x", and "H".

mms — An object of class multimarksetup

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

McClintock, B. T., Bailey, L. L., Dreher, B. P., and Link, W. A. 2014. Probit models for capture-recapture data subject to imperfect detection, individual heterogeneity and misidentification. *The Annals of Applied Statistics* 8: 2461-2484.

See Also

bobcat, processdata, multimodelClosed

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Run single chain using the default model for bobcat data
bobcat.dot<-multimarkClosed(bobcat)

#Posterior summary for monitored parameters
summary(bobcat.dot$mcmc)
plot(bobcat.dot$mcmc)
```

`multimarkClosedSCR` *Fit spatially-explicit population abundance models for capture-mark-recapture data consisting of multiple non-invasive marks*

Description

This function fits spatially-explicit population abundance models for capture-mark-recapture data consisting of multiple non-invasive marks using Bayesian analysis methods. Markov chain Monte Carlo (MCMC) is used to draw samples from the joint posterior distribution.

Usage

```
multimarkClosedSCR(
  Enc.Mat,
  trapCoords,
  studyArea = NULL,
  buffer = NULL,
  ncells = 1024,
  data.type = "never",
  covs = data.frame(),
  mms = NULL,
```

```
  mod.p = ~1,
  mod.delta = ~type,
  detection = "half-normal",
  parms = c("pbeta", "delta", "N"),
  nchains = 1,
  iter = 12000,
  adapt = 1000,
  bin = 50,
  thin = 1,
  burnin = 2000,
  taccept = 0.44,
  tuneadjust = 0.95,
  proppbeta = 0.1,
  propsigma = 1,
  propcenter = NULL,
  maxnumbasis = 1,
  a0delta = 1,
  a0alpha = 1,
  b0alpha = 1,
  sigma_bounds = NULL,
  mu0 = 0,
  sigma2_mu0 = 1.75,
  a0psi = 1,
  b0psi = 1,
  initial.values = NULL,
  known = integer(),
  scalemax = 10,
  printlog = FALSE,
  ...
)
```

Arguments

Enc.Mat	A matrix containing the observed encounter histories with rows corresponding to individuals and (ntraps*noccas) columns corresponding to traps and sampling occasions. The first noccas columns correspond to trap 1, the second noccas columns coresopond to trap 2, etc. Ignored unless mms=NULL.
trapCoords	A matrix of dimension ntraps x (2 + noccas) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate ("x"), and the second column the y-coordinate ("y"). The last noccas columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating. Ignored unless mms=NULL.
studyArea	is a 3-column matrix containing the coordinates for the centroids of a contiguous grid of cells that define the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns ("x" and "y") indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column ("avail") indicates whether the cell is available habitat (=1) or not (=0). All cells

must be square and have the same resolution. If studyArea=NULL (the default) and mms=NULL, then a square study area grid composed of ncells cells of available habitat is drawn around the bounding box of trapCoords based on buffer. Ignored unless mms=NULL. Note that rows should be ordered in raster cell order (raster cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom).

buffer — A scaler in same units as trapCoords indicating the buffer around the bounding box of trapCoords for defining the study area when studyArea=NULL. Ignored unless studyArea=NULL and mms=NULL.

ncells — The number of grid cells in the study area when studyArea=NULL. The square root of ncells must be a whole number. Default is ncells=1024. Ignored unless studyArea=NULL and mms=NULL.

data.type — Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted:

data.type="never" indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See bobcat. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3).

data.type="sometimes" indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

data.type="always" indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

covs — A data frame of time- and/or trap-dependent covariates for detection probabilities (ignored unless mms=NULL). The number of rows in the data frame must equal the number of traps times the number of sampling occasions (ntraps*noccas), where the first noccas rows correspond to trap 1, the second noccas rows correspond to trap 2, etc. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying mod.p and mod.phi.

mms	An optional object of class multimarkSCRsetup-class; if NULL it is created. See processdataSCR.
mod.p	Model formula for detection probability as a function of distance from activity centers. For example, mod.p=~1 specifies no effects (i.e., intercept only) other than distance, mod.p~time specifies temporal effects, mod.p~c specifies behavioral reponse (i.e., trap "happy" or "shy"), mod.p~trap specifies trap effects, and mod.p~time+c specifies additive temporal and behavioral effects.
mod.delta	Model formula for conditional probabilities of type 1 (delta_1) and type 2 (delta_2) encounters, given detection. Currently only mod.delta=~1 (i.e., $\delta_1 = \delta_2$) and mod.delta=~type (i.e., $\delta_1 \neq \delta_2$) are implemented.
detection	Model for detection probability as a function of distance from activity centers . Must be "half-normal" (of the form $\exp(-d^2/(2 * \sigma^2))$, where d is distance) or "exponential" (of the form $\exp(-d/\lambda)$).
parms	A character vector giving the names of the parameters and latent variables to monitor. Possible parameters are cloglog-scale detection probability parameters ("pbeta"), population abundance ("N"), conditional probability of type 1 or type 2 encounter, given detection ("delta)", probability of simultaneous type 1 and type 2 detection, given both types encountered ("alpha"), cloglog-scale distance term for the detection function ("sigma2_scr" when detection=``half-normal'' or "lambda" when detection=``exponential''), and the probability that a randomly selected individual from the M = nrow(Enc.Mat) observed individuals belongs to the n unique individuals encountered at least once ("psi"). Individual activity centers ("centers"), encounter history indices ("H"), and the log posterior density ("logPosterior") may also be monitored. Setting parms="all" monitors all possible parameters and latent variables.
nchains	The number of parallel MCMC chains for the model.
iter	The number of MCMC iterations.
adapt	The number of iterations for proposal distribution adaptation. If adapt = 0 then no adaptation occurs.
bin	Bin length for calculating acceptance rates during adaptive phase (0 < bin <= iter).
thin	Thinning interval for monitored parameters.
burnin	Number of burn-in iterations (0 <= burnin < iter).
taccept	Target acceptance rate during adaptive phase (0 < taccept <= 1). Acceptance rate is monitored every bin iterations. Default is taccept = 0.44.
tuneadjust	Adjustment term during adaptive phase (0 < tuneadjust <= 1). If acceptance rate is less than taccept, then proposal term (proppbeta or propsigma) is multiplied by tuneadjust. If acceptance rate is greater than or equal to taccept, then proposal term is divided by tuneadjust. Default is tuneadjust = 0.95.
proppbeta	Scaler or vector (of length k) specifying the initial standard deviation of the Normal(pbeta[j], proppbeta[j]) proposal distribution. If proppbeta is a scaler, then this value is used for all j = 1, ..., k. Default is proppbeta = 0.1.
propsigma	Scaler specifying the initial Gamma(shape = 1/propsigma, scale = sigma_scr * propsigma) proposal distribution for sigma_scr = sqrt(sigma2_scr) (or sqrt(lambda) = lambda if detection=``exponential''). Default is propsigma=1.

`propcenter` Scaler specifying the neighborhood distance when proposing updates to activity centers. When `propcenter=NULL` (the default), then propcenter = a*10, where a is the cell size for the study area grid, and each cell has (at most) approximately 300 neighbors.

`maxnumbasis` Maximum number of basis vectors to use when proposing latent history frequency updates. Default is `maxnumbasis = 1`, but higher values can potentially improve mixing.

`a0delta` Scaler or vector (of length d) specifying the prior for the conditional (on detection) probability of type 1 (delta_1), type 2 (delta_2), and both type 1 and type 2 encounters (1-delta_1-delta_2). If `a0delta` is a scaler, then this value is used for all a0delta[j] for j = 1, ..., d. For `mod.delta=~type`, d=3 with [delta_1, delta_2, 1-delta_1-delta_2] ~ Dirichlet(a0delta) prior. For `mod.delta=~1`, d=2 with [tau] ~ Beta(a0delta[1],a0delta[2]) prior, where (delta_1,delta_2,1-delta_1-delta_2) = (tau/2,tau/2,1-tau). See McClintock et al. (2013) for more details.

`a0alpha` Specifies "shape1" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `a0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`b0alpha` Specifies "shape2" parameter for [alpha] ~ Beta(a0alpha, b0alpha) prior. Only applicable when `data.type = "sometimes"`. Default is `b0alpha = 1`. Note that when `a0alpha = 1` and `b0alpha = 1`, then [alpha] ~ Unif(0,1).

`sigma_bounds` Positive vector of length 2 for the lower and upper bounds for the [sigma_scr] ~ Uniform(sigma_bounds[1], sigma_bounds[2]) (or [sqrt(lambda)] when `detection=``exponential''`) prior for the detection function term sigma_scr = sqrt(sigma2_scr) (or sqrt(lambda)).
When `sigma_bounds = NULL` (the default), then `sigma_bounds = c(1.e-6,max(diff(range(studyArea`

`mu0` Scaler or vector (of length k) specifying mean of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If `mu0` is a scaler, then this value is used for all j = 1, ..., k. Default is `mu0 = 0`.

`sigma2_mu0` Scaler or vector (of length k) specifying variance of pbeta[j] ~ Normal(mu0[j], sigma2_mu0[j]) prior. If `sigma2_mu0` is a scaler, then this value is used for all j = 1, ..., k. Default is `sigma2_mu0 = 1.75`.

`a0psi` Specifies "shape1" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `a0psi = 1`.

`b0psi` Specifies "shape2" parameter for [psi] ~ Beta(a0psi,b0psi) prior. Default is `b0psi = 1`.

`initial.values` Optional list of `nchain` list(s) specifying initial values for parameters and latent variables. Default is `initial.values = NULL`, which causes initial values to be generated automatically. In addition to the parameters ("`pbeta`", "`N`", "`delta_1`", "`delta_2`", "`alpha`", "`sigma2_scr`", "`centers`", and "`psi`"), initial values can be specified for the initial latent history frequencies ("`x`") and initial individual encounter history indices ("`H`").

`known` Optional integer vector indicating whether the encounter history of an individual is known with certainty (i.e., the observed encounter history is the true encounter history). Encounter histories with at least one type 4 encounter are automatically assumed to be known, and `known` does not need to be specified unless there exist encounter histories that do not contain a type 4 encounter that happen to be

known with certainty (e.g., from independent telemetry studies). If specified, known = c(v_1,v_2,...,v_M) must be a vector of length M = nrow(Enc.Mat) where v_i = 1 if the encounter history for individual i is known (v_i = 0 otherwise). Note that known all-zero encounter histories (e.g., '000') are ignored.

scalemax — Upper bound for internal re-scaling of grid cell centroid coordinates. Default is scalemax=10, which re-scales the centroids to be between 0 and 10. Re-scaling is done internally to avoid numerical overflows during model fitting. Ignored unless mms=NULL.

printlog — Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.

... — Additional "parameters" arguments for specifying mod.p. See make.design.data.

Details

The first time multimarkSCRClosed is called, it will likely produce a firewall warning alerting users that R has requested the ability to accept incoming network connections. Incoming network connections are required to use parallel processing as implemented in multimarkClosed. Note that setting parms="all" is required for any multimarkClosed model output to be used in multimodelClosed.

Value

A list containing the following:

mcmc — Markov chain Monte Carlo object of class mcmc.list.

mod.p — Model formula for detection probability (as specified by mod.p above).

mod.delta — Model formula for conditional probability of type 1 or type 2 encounter, given detection (as specified by mod.delta above).

mod.det — Model formula for detection function (as specified by detection above).

DM — A list of design matrices for detection probability generated for model mod.p, where DM$p is the design matrix for initial capture probability (p) and DM$c is the design matrix for recapture probability (c).

initial.values — A list containing the parameter and latent variable values at iteration iter for each chain. Values are provided for "pbeta", "N", "delta_1", "delta_2", "alpha", "sigma2_scr", "centers", "psi", "x", and "H".

mms — An object of class multimarkSCRsetup

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

Gopalaswamy, A.M., Royle, J.A., Hines, J.E., Singh, P., Jathanna, D., Kumar, N. and Karanth, K.U. 2012. Program SPACECAP: software for estimating animal density using spatially explicit capture-recapture models. *Methods in Ecology and Evolution* 3:1067-1072.

King, R., McClintock, B. T., Kidney, D., and Borchers, D. L. 2016. Capture-recapture abundance estimation using a semi-complete data likelihood approach. *The Annals of Applied Statistics* 10: 264-285

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

McClintock, B. T., Bailey, L. L., Dreher, B. P., and Link, W. A. 2014. Probit models for capture-recapture data subject to imperfect detection, individual heterogeneity and misidentification. *The Annals of Applied Statistics* 8: 2461-2484.

Royle, J.A., Karanth, K.U., Gopalaswamy, A.M. and Kumar, N.S. 2009. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90: 3233-3244.

See Also

processdataSCR.

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarkSCRsetup" from simulated data
sim.data<-simdataClosedSCR()
Enc.Mat <- sim.data$Enc.Mat
trapCoords <- sim.data$spatialInputs$trapCoords
studyArea <- sim.data$spatialInputs$studyArea

#Run single chain using the default model for simulated data
example.dot<-multimarkClosedSCR(Enc.Mat,trapCoords,studyArea)

#Posterior summary for monitored parameters
summary(example.dot$mcmc)
plot(example.dot$mcmc)
```

---

multimarkSCRsetup-class

Class "multimarkSCRsetup"

Description

A class of spatial 'mulitmark' model inputs

Slots

`Enc.Mat` Object of class `"matrix"`. The observed encounter histories (with rows corresponding to individuals and columns corresponding to sampling occasions).

`data.type` Object of class `"character"`. The encounter history data type ("never", "sometimes", or "always").

`vAll.hists` Object of class `"integer"`. An ordered vector containing all possible encounter histories in sequence.

`Aprime` Object of class `"sparseMatrix"`. Transpose of the A matrix mapping latent encounter histories to observed histories.

`indBasis` Object of class `"numeric"`.An ordered vector of the indices of the three encounter histories updated by each basis vector.

`ncolbasis` Object of class `"integer"`. The number of needed basis vectors.

`knownx` Object of class `"integer"`. Frequencies of known encounter histories.

`C` Object of class `"integer"`. Sampling occasion of first capture for each encounter history.

`L` Object of class `"integer"`. Sampling occasion of last capture for each encounter history.

`naivex` Object of class `"integer"`. "Naive" latent history frequencies assuming a one-to-one mapping with `Enc.Mat`.

`covs` Object of class `"data.frame"`. Temporal covariates for detection probability (the number of rows in the data frame must equal the number of sampling occasions).

`spatialInputs` Object of class `"list"`. List is of length 4 containing `trapCoords` and `studyArea` after re-scaling coordinates based on `maxscale`, as well as the original (not re-scaled) grid cell resolution (`origCellRes`) and re-scaling range (`Srange`).

Objects from the Class

Objects can be created by calls of the form `processdata(Enc.Mat, ...)` or `new("multimarkSCRsetup", ...)`.

Methods

No methods defined with class "multimarkSCRsetup".

Author(s)

Brett T. McClintock

See Also

`processdataSCR`

Examples

```
showClass("multimarkSCRsetup")
```

multimarksetup-class *Class* "multimarksetup"

Description

A class of 'mulitmark' model inputs

Slots

`Enc.Mat` Object of class "matrix". The observed encounter histories (with rows corresponding to individuals and columns corresponding to sampling occasions).

`data.type` Object of class "character". The encounter history data type ("never", "sometimes", or "always").

`vAll.hists` Object of class "integer". An ordered vector containing all possible encounter histories in sequence.

`Aprime` Object of class "sparseMatrix". Transpose of the A matrix mapping latent encounter histories to observed histories.

`indBasis` Object of class "numeric".An ordered vector of the indices of the three encounter histories updated by each basis vector.

`ncolbasis` Object of class "integer". The number of needed basis vectors.

`knownx` Object of class "integer". Frequencies of known encounter histories.

`C` Object of class "integer". Sampling occasion of first capture for each encounter history.

`L` Object of class "integer". Sampling occasion of last capture for each encounter history.

`naivex` Object of class "integer". "Naive" latent history frequencies assuming a one-to-one mapping with `Enc.Mat`.

`covs` Object of class "data.frame". Temporal covariates for detection probability (the number of rows in the data frame must equal the number of sampling occasions).

Objects from the Class

Objects can be created by calls of the form `processdata(Enc.Mat, ...)` or `new("multimarksetup", ...)`.

Methods

No methods defined with class "multimarksetup".

Author(s)

Brett T. McClintock

See Also

processdata

Examples

```
showClass("multimarksetup")
```

`multimodelCJS` *Multimodel inference for 'multimark' open population survival models*

Description

This function performs Bayesian multimodel inference for a set of 'multimark' open population survival (i.e., Cormack-Jolly-Seber) models using the reversible jump Markov chain Monte Carlo (RJMCMC) algorithm proposed by Barker & Link (2013).

Usage

```
multimodelCJS(
  modlist,
  modprior = rep(1/length(modlist), length(modlist)),
  monparms = "phi",
  miter = NULL,
  mburnin = 0,
  mthin = 1,
  M1 = NULL,
  pbetapropsd = 1,
  zppropsd = NULL,
  phibetapropsd = 1,
  zphipropsd = NULL,
  sigppropshape = 1,
  sigppropscale = 0.01,
  sigphipropshape = 1,
  sigphipropscale = 0.01,
  printlog = FALSE
)
```

Arguments

`modlist`	A list of individual model output lists returned by `multimarkCJS`. The models must have the same number of chains and MCMC iterations.
`modprior`	Vector of length `length(modlist)` containing prior model probabilities. Default is `modprior = rep(1/length(modlist), length(modlist))`.
`monparms`	Parameters to monitor. Only parameters common to all models can be monitored (e.g., `"pbeta[(Intercept)]"`, `"phibeta[(Intercept)]"`, `"psi"`), but derived survival (`"phi"`) and capture (`"p"`) probabilities can also be monitored. Default is `monparms = "phi"`.
`miter`	The number of RJMCMC iterations per chain. If `NULL`, then the number of MCMC iterations for each individual model chain is used.
`mburnin`	Number of burn-in iterations (`0 <= mburnin < miter`).

mthin — Thinning interval for monitored parameters.

M1 — Integer vector indicating the initial model for each chain, where M1_j=i initializes the RJMCMC algorithm for chain j in the model corresponding to modlist[[i]] for i=1,..., length(modlist). If NULL, the algorithm for all chains is initialized in the most general model. Default is M1=NULL.

pbetapropsd — Scaler specifying the standard deviation of the Normal(0, pbetapropsd) proposal distribution for "pbeta" parameters. Default is pbetapropsd=1. See Barker & Link (2013) for more details.

zppropsd — Scaler specifying the standard deviation of the Normal(0, zppropsd) proposal distribution for "zp" parameters. Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. If NULL, zppropsd = sqrt(sigma2_zp) is used. Default is zppropsd=NULL. See Barker & Link (2013) for more details.

phibetapropsd — Scaler specifying the standard deviation of the Normal(0, phibetapropsd) proposal distribution for "phibeta" parameters. Default is phibetapropsd=1. See Barker & Link (2013) for more details.

zphipropsd — Scaler specifying the standard deviation of the Normal(0, zphipropsd) proposal distribution for "zphi" parameters. Only applies if at least one (but not all) model(s) include individual hetergeneity in survival probability. If NULL, zphipropsd = sqrt(sigma2_zphi) is used. Default is zphipropsd=NULL. See Barker & Link (2013) for more details.

sigppropshape — Scaler specifying the shape parameter of the invGamma(shape = sigppropshape, scale = sigppropscale) proposal distribution for "sigma2_zp". Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. Default is sigppropshape=1. See Barker & Link (2013) for more details.

sigppropscale — Scaler specifying the scale parameter of the invGamma(shape = sigppropshape, scale = sigppropscale) proposal distribution for "sigma2_zp". Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. Default is sigppropscale=0.01. See Barker & Link (2013) for more details.

sigphipropshape — Scaler specifying the shape parameter of the invGamma(shape = sigphipropshape, scale = sigphipropscale) proposal distribution for "sigma2_zphi". Only applies if at least one (but not all) model(s) include individual hetergeneity in survival probability. Default is sigphipropshape=1. See Barker & Link (2013) for more details.

sigphipropscale — Scaler specifying the scale parameter of the invGamma(shape = sigphipropshape, scale = sigphipropscale) proposal distribution for "sigma_zphi". Only applies if at least one (but not all) model(s) include individual hetergeneity in survival probability. Default is sigphipropscale=0.01. See Barker & Link (2013) for more details.

printlog — Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains,

setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.

Details

Note that setting parms="all" is required when fitting individual multimarkCJS models to be included in modlist.

Value

A list containing the following:

rjmcmc	Reversible jump Markov chain Monte Carlo object of class mcmc.list. Includes RJMCMC output for monitored parameters and the current model at each iteration ("M").
pos.prob	A list of calculated posterior model probabilities for each chain, including the overall posterior model probabilities across all chains.

Author(s)

Brett T. McClintock

References

Barker, R. J. and Link. W. A. 2013. Bayesian multimodel inference by RJMCMC: a Gibbs sampling approach. The American Statistician 67: 150-156.

See Also

multimarkCJS, processdata

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarksetup" from simulated data
data_type = "always"
noccas <- 5
phibetaTime <- seq(2,0,length=noccas-1) # declining trend in survival
data <- simdataCJS(noccas=5,phibeta=phibetaTime,data.type=data_type)
setup <- processdata(data$Enc.Mat,data.type=data_type)

#Run single chain using the default model. Note parms="all".
sim.pdot.phidot <- multimarkCJS(mms=setup,parms="all",iter=1000,adapt=500,burnin=500)

#Run single chain with temporal trend for phi. Note parms="all".
sim.pdot.phiTime <- multimarkCJS(mms=setup,mod.phi=~Time,parms="all",iter=1000,adapt=500,burnin=500)
```

```
#Perform RJMCMC using defaults
modlist <- list(mod1=sim.pdot.phidot,mod2=sim.pdot.phiTime)
sim.M <- multimodelCJS(modlist=modlist)

#Posterior model probabilities
sim.M$pos.prob

#multimodel posterior summary for survival (display first cohort only)
summary(sim.M$rjmcmc[,paste0("phi[1,",1:(noccas-1),"]")])
```

`multimodelClosed` *Multimodel inference for 'multimark' closed population abundance models*

Description

This function performs Bayesian multimodel inference for a set of 'multimark' closed population abundance models using the reversible jump Markov chain Monte Carlo (RJMCMC) algorithm proposed by Barker & Link (2013).

Usage

```
multimodelClosed(
  modlist,
  modprior = rep(1/length(modlist), length(modlist)),
  monparms = "N",
  miter = NULL,
  mburnin = 0,
  mthin = 1,
  M1 = NULL,
  pbetapropsd = 1,
  zppropsd = NULL,
  sigppropshape = 6,
  sigppropscale = 4,
  printlog = FALSE
)
```

Arguments

`modlist`	A list of individual model output lists returned by `multimarkClosed` or `markClosed`. The models must have the same number of chains and MCMC iterations.
`modprior`	Vector of length `length(modlist)` containing prior model probabilities. Default is `modprior = rep(1/length(modlist), length(modlist))`.
`monparms`	Parameters to monitor. Only parameters common to all models can be monitored (e.g., `"pbeta[(Intercept)]"`, `"N"`), but derived capture (`"p"`) and recapture (`"c"`) probabilities can also be monitored. Default is `monparms = "N"`.

miter	The number of RJMCMC iterations per chain. If NULL, then the number of MCMC iterations for each individual model chain is used.
mburnin	Number of burn-in iterations (0 <= mburnin < miter).
mthin	Thinning interval for monitored parameters.
M1	Integer vector indicating the initial model for each chain, where M1_j=i initializes the RJMCMC algorithm for chain j in the model corresponding to modlist[[i]] for i=1,..., length(modlist). If NULL, the algorithm for all chains is initialized in the most general model. Default is M1=NULL.
pbetapropsd	Scaler specifying the standard deviation of the Normal(0, pbetapropsd) proposal distribution for "pbeta" parameters. Default is pbetapropsd=1. See Barker & Link (2013) for more details.
zppropsd	Scaler specifying the standard deviation of the Normal(0, zppropsd) proposal distribution for "zp" parameters. Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. If NULL, zppropsd = sqrt(sigma2_zp) is used. Default is zppropsd=NULL. See Barker & Link (2013) for more details.
sigppropshape	Scaler specifying the shape parameter of the invGamma(shape = sigppropshape, scale = sigpropscale) proposal distribution for sigma_zp. Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. Default is sigppropshape=6. See Barker & Link (2013) for more details.
sigppropscale	Scaler specifying the scale parameter of the invGamma(shape = sigppropshape, scale = sigpropscale) proposal distribution for sigma_zp. Only applies if at least one (but not all) model(s) include individual hetergeneity in detection probability. Default is sigppropscale=4. See Barker & Link (2013) for more details.
printlog	Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.

Details

Note that setting parms="all" is required when fitting individual multimarkClosed or markClosed models to be included in modlist.

Value

A list containing the following:

rjmcmc	Reversible jump Markov chain Monte Carlo object of class mcmc.list. Includes RJMCMC output for monitored parameters and the current model at each iteration ("M").
pos.prob	A list of calculated posterior model probabilities for each chain, including the overall posterior model probabilities across all chains.

Author(s)

Brett T. McClintock

References

Barker, R. J. and Link. W. A. 2013. Bayesian multimodel inference by RJMCMC: a Gibbs sampling approach. The American Statistician 67: 150-156.

See Also

multimarkClosed, markClosed, processdata

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarksetup"
setup <- processdata(bobcat)

#Run single chain using the default model for bobcat data. Note parms="all".
bobcat.dot <- multimarkClosed(mms=setup,parms="all",iter=1000,adapt=500,burnin=500)

#Run single chain for bobcat data with time effects. Note parms="all".
bobcat.time <- multimarkClosed(mms=setup,mod.p=~time,parms="all",iter=1000,adapt=500,burnin=500)

#Perform RJMCMC using defaults
modlist <- list(mod1=bobcat.dot,mod2=bobcat.time)
bobcat.M <- multimodelClosed(modlist=modlist,monparms=c("N","p"))

#Posterior model probabilities
bobcat.M$pos.prob

#multimodel posterior summary for abundance
summary(bobcat.M$rjmcmc[,"N"])
```

---

multimodelClosedSCR *Multimodel inference for 'multimark' spatial population abundance models*

Description

This function performs Bayesian multimodel inference for a set of 'multimark' spatial population abundance models using the reversible jump Markov chain Monte Carlo (RJMCMC) algorithm proposed by Barker & Link (2013).

Usage

```
multimodelClosedSCR(
  modlist,
  modprior = rep(1/length(modlist), length(modlist)),
  monparms = "N",
  miter = NULL,
  mburnin = 0,
  mthin = 1,
  M1 = NULL,
  pbetapropsd = 1,
  sigpropmean = 0.8,
  sigpropsd = 0.4,
  printlog = FALSE
)
```

Arguments

modlist	A list of individual model output lists returned by multimarkClosedSCR or markClosedSCR. The models must have the same number of chains and MCMC iterations.
modprior	Vector of length length(modlist) containing prior model probabilities. Default is modprior = rep(1/length(modlist), length(modlist)).
monparms	Parameters to monitor. Only parameters common to all models can be monitored (e.g., "pbeta[(Intercept)]", "N", "sigma2_scr"), but derived density ("D") as well as capture ("p") and recapture ("c") probabilities (at distance zero from activity centers) can also be monitored. Default is monparms = "N".
miter	The number of RJMCMC iterations per chain. If NULL, then the number of MCMC iterations for each individual model chain is used.
mburnin	Number of burn-in iterations (0 <= mburnin < miter).
mthin	Thinning interval for monitored parameters.
M1	Integer vector indicating the initial model for each chain, where M1_j=i initializes the RJMCMC algorithm for chain j in the model corresponding to modlist[[i]] for i=1,..., length(modlist). If NULL, the algorithm for all chains is initialized in the most general model. Default is M1=NULL.
pbetapropsd	Scaler specifying the standard deviation of the Normal(0, pbetapropsd) proposal distribution for "pbeta" parameters. Default is pbetapropsd=1. See Barker & Link (2013) for more details.
sigpropmean	Scaler specifying the mean of the inverse Gamma proposal distribution for sigma2_scr (or lambda if detection=``exponential''). Only applies if models do not have the same detection function (i.e., "half-normal" or "exponential"). Default is sigpropmean=0.8. See Barker & Link (2013) for more details.
sigpropsd	Scaler specifying the standard deviation of the inverse Gamma proposal distribution for sigma2_scr (or lambda if detection=``exponential''). Only applies if models do not have the same detection function (i.e., "half-normal" or "exponential"). Default is sigpropsd=0.4. See Barker & Link (2013) for more details.

printlog — Logical indicating whether to print the progress of chains and any errors to a log file in the working directory. Ignored when nchains=1. Updates are printed to log file as 1% increments of iter of each chain are completed. With >1 chains, setting printlog=TRUE is probably most useful for Windows users because progress and errors are automatically printed to the R console for "Unix-like" machines (i.e., Mac and Linux) when printlog=FALSE. Default is printlog=FALSE.

Details

Note that setting parms="all" is required when fitting individual multimarkClosedSCR or markClosedSCR models to be included in modlist.

Value

A list containing the following:

rjmcmc — Reversible jump Markov chain Monte Carlo object of class mcmc.list. Includes RJMCMC output for monitored parameters and the current model at each iteration ("M").

pos.prob — A list of calculated posterior model probabilities for each chain, including the overall posterior model probabilities across all chains.

Author(s)

Brett T. McClintock

References

Barker, R. J. and Link. W. A. 2013. Bayesian multimodel inference by RJMCMC: a Gibbs sampling approach. The American Statistician 67: 150-156.

See Also

multimarkClosedSCR, processdataSCR

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarkSCRsetup"
sim.data<-simdataClosedSCR()
Enc.Mat<-sim.data$Enc.Mat
trapCoords<-sim.data$spatialInputs$trapCoords
studyArea<-sim.data$spatialInputs$studyArea
setup<-processdataSCR(Enc.Mat,trapCoords,studyArea)

#Run single chain using the default model for simulated data. Note parms="all".
example.dot <- multimarkClosedSCR(mms=setup,parms="all",iter=1000,adapt=500,burnin=500)

#Run single chain for simulated data with behavior effects. Note parms="all".
```

```
example.c <- multimarkClosedSCR(mms=setup,mod.p=~c,parms="all",iter=1000,adapt=500,burnin=500)

#Perform RJMCMC using defaults
modlist <- list(mod1=example.dot,mod2=example.c)
example.M <- multimodelClosedSCR(modlist=modlist,monparms=c("N","D","sigma2_scr"))

#Posterior model probabilities
example.M$pos.prob

#multimodel posterior summary for abundance and density
summary(example.M$rjmcmc[,c("N","D")])
```

`plotSpatialData` *Plot spatial capture-mark-recapture data*

Description

This function plots the study area grid, available habitat, and trap coordinates for spatial capture-recapture studies. Activity centers and capture locations can also be plotted.

Usage

```
plotSpatialData(
  mms = NULL,
  trapCoords,
  studyArea,
  centers = NULL,
  trapLines = FALSE
)
```

Arguments

`mms`	An optional object of class `multimarkSCRsetup-class` from which the (re-scaled) study area and trap coordinates are plotted.
`trapCoords`	A matrix of dimension `ntraps` x (2 + `noccas`) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate, and the second column the y-coordinate. The last `noccas` columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating. Ignored unless `mms=NULL`.
`studyArea`	A 3-column matrix defining the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column indicates whether the cell is available habitat (=1) or not (=0). All cells must have the same resolution. Ignored unless `mms=NULL`. Note that rows should be ordered in raster cell order (raster cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom).

centers — An optional vector indicating the grid cell (i.e., the row of studyArea) that contains the true (latent) activity centers for each individual. If mms is provided, then centers must be of length nrow(Enc.Mat) (i.e., a center must be provided for each observed individual).

trapLines — Logical indicating whether to draw lines from activity centers to respective traps at which each individual was captured. Default is trapLines=FALSE. Ignored when mms=NULL or centers=NULL.

Author(s)

Brett T. McClintock

Examples

```
#Plot the tiger example data
plotSpatialData(trapCoords=tiger$trapCoords,studyArea=tiger$studyArea)
```

---

processdata — *Generate model inputs for fitting 'multimark' models*

Description

This function generates an object of class multimarksetup that is required to fit 'multimark' models.

Usage

```
processdata(
  Enc.Mat,
  data.type = "never",
  covs = data.frame(),
  known = integer()
)
```

Arguments

Enc.Mat — A matrix of observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions (ignored unless mms=NULL).

data.type — Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted:

data.type="never" indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only

left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See bobcat. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3).

`data.type="sometimes"` indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

`data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`covs` A data frame of temporal covariates for detection probabilities (ignored unless `mms=NULL`). The number of rows in the data frame must equal the number of sampling occasions. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying `mod.p` and `mod.phi`.

`known` Optional integer vector indicating whether the encounter history of an individual is known with certainty (i.e., the observed encounter history is the true encounter history). Encounter histories with at least one type 4 encounter are automatically assumed to be known, and `known` does not need to be specified unless there exist encounter histories that do not contain a type 4 encounter that happen to be known with certainty (e.g., from independent telemetry studies). If specified, `known = c(v_1,v_2,...,v_M)` must be a vector of length `M = nrow(Enc.Mat)` where `v_i = 1` if the encounter history for individual `i` is known (`v_i = 0` otherwise). Note that known all-zero encounter histories (e.g., '000') are ignored.

Value

An object of class `multimarksetup`.

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

See Also

multimarksetup-class, multimarkClosed, bobcat

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarksetup"
setup <- processdata(bobcat)

#Run single chain using the default model for bobcat data
bobcat.dot<-multimarkClosed(mms=setup)

#Run single chain for bobcat data with temporal effects (i.e., mod.p=~time)
bobcat.time <- multimarkClosed(mms=setup,mod.p=~time)
```

---

processdataSCR *Generate model inputs for fitting spatial 'multimark' models*

Description

This function generates an object of class multimarkSCRsetup that is required to fit spatial 'multimark' models.

Usage

```
processdataSCR(
  Enc.Mat,
  trapCoords,
  studyArea = NULL,
  buffer = NULL,
  ncells = NULL,
  data.type = "never",
  covs = data.frame(),
  known = integer(),
  scalemax = 10
)
```

Arguments

Enc.Mat	A matrix containing the observed encounter histories with rows corresponding to individuals and (ntraps*noccas) columns corresponding to traps and sampling occasions. The first noccas columns correspond to trap 1, the second noccas columns coresopond to trap 2, etc. Ignored unless mms=NULL.

`trapCoords` A matrix of dimension `ntraps` x (2 + `noccas`) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate, and the second column the y-coordinate. The last `noccas` columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating.

`studyArea` is a 3-column matrix containing the coordinates for the centroids of a contiguous grid of cells that define the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column indicates whether the cell is available habitat (=1) or not (=0). All cells must be square and have the same resolution. If `studyArea=NULL` (the default), then a square study area grid composed of `ncells` cells of available habitat is drawn around the bounding box of `trapCoords` based on `buffer`. Note that rows should be ordered in raster cell order (raster cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom).

`buffer` A scaler in same units as `trapCoords` indicating the buffer around the bounding box of `trapCoords` for defining the study area when `studyArea=NULL`. Ignored unless `studyArea=NULL`.

`ncells` The number of grid cells in the study area when `studyArea=NULL`. The square root of `ncells` must be a whole number. Default is `ncells=1024`. Ignored unless `studyArea=NULL`.

`data.type` Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted:

`data.type="never"` indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See `bobcat`. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3).

`data.type="sometimes"` indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

`data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent

encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`covs` A data frame of time- and/or trap-dependent covariates for detection probabilities (ignored unless `mms=NULL`). The number of rows in the data frame must equal the number of traps times the number of sampling occasions (`ntraps*noccas`), where the first `noccas` rows correspond to trap 1, the second `noccas` rows correspond to trap 2, etc. Covariate names cannot be "time", "age", or "h"; these names are reserved for temporal, behavioral, and individual effects when specifying `mod.p` and `mod.phi`.

`known` Optional integer vector indicating whether the encounter history of an individual is known with certainty (i.e., the observed encounter history is the true encounter history). Encounter histories with at least one type 4 encounter are automatically assumed to be known, and `known` does not need to be specified unless there exist encounter histories that do not contain a type 4 encounter that happen to be known with certainty (e.g., from independent telemetry studies). If specified, `known = c(v_1,v_2,...,v_M)` must be a vector of length `M = nrow(Enc.Mat)` where `v_i = 1` if the encounter history for individual `i` is known (`v_i = 0` otherwise). Note that known all-zero encounter histories (e.g., '000') are ignored.

`scalemax` Upper bound for internal re-scaling of grid cell centroid coordinates. Default is `scalemax=10`, which re-scales the centroids to be between 0 and 10. Re-scaling is done internally to avoid numerical overflows during model fitting.

Value

An object of class `multimarkSCRsetup`.

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

Gopalaswamy, A.M., Royle, J.A., Hines, J.E., Singh, P., Jathanna, D., Kumar, N. and Karanth, K.U. 2012. Program SPACECAP: software for estimating animal density using spatially explicit capture-recapture models. *Methods in Ecology and Evolution* 3:1067-1072.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

Royle, J.A., Karanth, K.U., Gopalaswamy, A.M. and Kumar, N.S. 2009. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90: 3233-3244.

See Also

`multimarkSCRsetup-class`, `multimarkClosedSCR`

Examples

```
# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

#Generate object of class "multimarksetup" from simulated data
sim.data<-simdataClosedSCR()
Enc.Mat <- sim.data$Enc.Mat
trapCoords <- sim.data$spatialInputs$trapCoords
studyArea <- sim.data$spatialInputs$studyArea
setup <- processdataSCR(Enc.Mat,trapCoords,studyArea)

#Run single chain using the default model for simulated data
example.dot<-multimarkClosedSCR(mms=setup)
```

`simdataCJS` *Simulate open population capture-mark-recapture data arising from multiple non-invasive marks*

Description

This function generates encounter histories from simulated open population capture-mark-recapture data consisting of multiple non-invasive marks.

Usage

```
simdataCJS(
  N = 100,
  noccas = 5,
  pbeta = -0.25,
  sigma2_zp = 0,
  phibeta = 1.6,
  sigma2_zphi = 0,
  delta_1 = 0.4,
  delta_2 = 0.4,
  alpha = 0.5,
  data.type = "never",
  link = "probit"
)
```

Arguments

`N`	Number of individuals.
`noccas`	Number of sampling occasions. `floor(N/noccas)` individuals are first encountered on each occasion.
`pbeta`	Logit- or probit-scale intercept term(s) for capture probability (p). Must be a scaler or vector of length `noccas`.

`sigma2_zp` Logit- or probit-scale individual heterogeneity variance term for capture probability (p).

`phibeta` Logit- or probit-scale intercept term(s) for survival probability (ϕ). Must be a scaler or vector of length `noccas`.

`sigma2_zphi` Logit- or probit-scale individual heterogeneity variance term for survival probability (ϕ).

`delta_1` Conditional probability of type 1 encounter, given detection.

`delta_2` Conditional probability of type 2 encounter, given detection.

`alpha` Conditional probability of simultaneous type 1 and type 2 detection, given both types encountered. Only applies when `data.type="sometimes"`.

`data.type` Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted:

`data.type="never"` indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See `bobcat`. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3).

`data.type="sometimes"` indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

`data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`link` Link function for detection probability. Must be "`logit`" or "`probit`". Note that `multimarkCJS` is currently implemented for the probit link only.

Value

A list containing the following:

`Enc.Mat` A matrix containing the observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions.

`trueEnc.Mat` A matrix containing the true (latent) encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions.

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

See Also

processdata, multimarkCJS

Examples

```
#simulate data for data.type="sometimes" using defaults
data<-simdataCJS(data.type="sometimes")
```

`simdataClosed` *Simulate closed population capture-mark-recapture data arising from multiple non-invasive marks*

Description

This function generates encounter histories from simulated closed population capture-mark-recapture data consisting of multiple non-invasive marks.

Usage

```
simdataClosed(
  N = 100,
  noccas = 5,
  pbeta = -0.4,
  tau = 0,
  sigma2_zp = 0,
  delta_1 = 0.4,
  delta_2 = 0.4,
  alpha = 0.5,
  data.type = "never",
  link = "logit"
)
```

Arguments

N	True population size or abundance.
noccas	The number of sampling occasions.
pbeta	Logit- or probit-scale intercept term(s) for capture probability (p). Must be a scaler or vector of length noccas.
tau	Additive logit- or probit-scale behavioral effect term for recapture probability (c).
sigma2_zp	Logit- or probit-scale individual heterogeneity variance term.
delta_1	Conditional probability of type 1 encounter, given detection.
delta_2	Conditional probability of type 2 encounter, given detection.
alpha	Conditional probability of simultaneous type 1 and type 2 detection, given both types encountered. Only applies when data.type="sometimes".
data.type	Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted: data.type="never" indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See bobcat. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3). data.type="sometimes" indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4). data.type="always" indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).
link	Link function for detection probability. Must be "logit" or "probit". Note that multimarkClosed is currently implemented for the logit link only.

Value

A list containing the following:

Enc.Mat	A matrix containing the observed encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions.
trueEnc.Mat	A matrix containing the true (latent) encounter histories with rows corresponding to individuals and columns corresponding to sampling occasions.

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

See Also

processdata, multimarkClosed

Examples

```
#simulate data for data.type="sometimes" using defaults
data<-simdataClosed(data.type="sometimes")
```

---

simdataClosedSCR — *Simulate spatially-explicit capture-mark-recapture data from a (demographically) closed population with multiple non-invasive marks*

Description

This function generates encounter histories from spatially-explicit capture-mark-recapture data consisting of multiple non-invasive marks.

Usage

```
simdataClosedSCR(
  N = 30,
  ntraps = 9,
  noccas = 5,
  pbeta = 0.25,
  tau = 0,
  sigma2_scr = 0.75,
  lambda = 0.75,
  delta_1 = 0.4,
  delta_2 = 0.4,
```

```
  alpha = 0.5,
  data.type = "never",
  detection = "half-normal",
  spatialInputs = NULL,
  buffer = 3 * sqrt(sigma2_scr),
  ncells = 1024,
  scalemax = 10,
  plot = TRUE
)
```

Arguments

N	True population size or abundance.
ntraps	The number of traps. If trapCoords=NULL, the square root of ntraps must be a whole number in order to create a regular grid of trap coordinates on a square.
noccas	Scaler indicating the number of sampling occasions per trap.
pbeta	Complementary loglog-scale intercept term for detection probability (p). Must be a scaler or vector of length noccas.
tau	Additive complementary loglog-scale behavioral effect term for recapture probability (c).
sigma2_scr	Complementary loglog-scale term for effect of distance in the "half-normal" detection function. Ignored unless detection=``half-normal''.
lambda	Complementary loglog-scale term for effect of distance in the "exponential" detection function. Ignored unless detection=``exponential''.
delta_1	Conditional probability of type 1 encounter, given detection.
delta_2	Conditional probability of type 2 encounter, given detection.
alpha	Conditional probability of simultaneous type 1 and type 2 detection, given both types encountered. Only applies when data.type="sometimes".
data.type	Specifies the encounter history data type. All data types include non-detections (type 0 encounter), type 1 encounter (e.g., left-side), and type 2 encounters (e.g., right-side). When both type 1 and type 2 encounters occur for the same individual within a sampling occasion, these can either be "non-simultaneous" (type 3 encounter) or "simultaneous" (type 4 encounter). Three data types are currently permitted: data.type="never" indicates both type 1 and type 2 encounters are never observed for the same individual within a sampling occasion, and observed encounter histories therefore include only type 1 or type 2 encounters (e.g., only left- and right-sided photographs were collected). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), and type 2 encounters (2). See bobcat. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 3 encounters (3). data.type="sometimes" indicates both type 1 and type 2 encounters are sometimes observed (e.g., both-sided photographs are sometimes obtained, but not necessarily for all individuals). Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters

(3), and type 4 encounters (4). Type 3 encounters can only be observed when an individual has at least one type 4 encounter. Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), type 3 encounters (3), and type 4 encounters (4).

`data.type="always"` indicates both type 1 and type 2 encounters are always observed, but some encounter histories may still include only type 1 or type 2 encounters. Observed encounter histories can consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4). Latent encounter histories consist of non-detections (0), type 1 encounters (1), type 2 encounters (2), and type 4 encounters (4).

`detection` Model for detection probability as a function of distance from activity centers. Must be "`half-normal`" (of the form $\exp(-d^2/(2*\sigma^2))$, where d is distance) or "`exponential`" (of the form $\exp(-d/\lambda)$).

`spatialInputs` A list of length 3 composed of objects named `trapCoords`, `studyArea`, and `centers`:

`trapCoords` is a matrix of dimension `ntraps` x (2 + `noccas`) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate ("x"), and the second column the y-coordinate ("y"). The last `noccas` columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating.

`studyArea` is a 3-column matrix defining the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns ("x" and "y") indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column ("avail") indicates whether the cell is available habitat (=1) or not (=0). All grid cells must have the same resolution. Note that rows should be ordered in raster cell order (raster cell numbers start at 1 in the upper left corner, and increase from left to right, and then from top to bottom).

`centers` is a `N`-vector indicating the grid cell (i.e., the row of `studyArea`) that contains the true (latent) activity centers for each individual in the population.

If `spatialInputs=NULL` (the default), then all traps are assumed to be operating on all occasions, the study area is assumed to be composed of `ncells` grid cells, grid cells within `buffer` of the trap array are assumed to be available habitat, and the activity centers are randomly assigned to grid cells of available habitat.

`buffer` A scaler indicating the buffer around the bounding box of `trapCoords` for defining the study area and available habitat when `spatialInputs=NULL`. Default is `buffer=3*sqrt(sigma2_scr)`. Ignored unless `spatialInputs=NULL`.

`ncells` The number of grid cells in the study area when `studyArea=NULL`. The square root of `ncells` must be a whole number. Default is `ncells=1024`. Ignored unless `spatialInputs=NULL`.

`scalemax` Upper bound for grid cell centroid x- and y-coordinates. Default is `scalemax=10`, which scales the x- and y-coordinates to be between 0 and 10. Ignored unless `spatialInputs=NULL`.

`plot` Logical indicating whether to plot the simulated trap coordinates, study area, and activity centers using `plotSpatialData`. Default is `plot=TRUE`

Details

Please be very careful when specifying your own spatialInputs; multimarkClosedSCR and markClosedSCR do little to verify that these make sense during model fitting.

Value

A list containing the following:

Enc.Mat	Matrix containing the observed encounter histories with rows corresponding to individuals and (ntraps*noccas) columns corresponding to traps and sampling occasions. The first noccas columns correspond to trap 1, the second noccas columns corresopond to trap 2, etc.
trueEnc.Mat	Matrix containing the true (latent) encounter histories with rows corresponding to individuals and (ntraps*noccas) columns corresponding to traps and sampling occasions. The first noccas columns correspond to trap 1, the second noccas columns corresopond to trap 2, etc.
spatialInputs	List of length 2 with objects named trapCoords and studyArea: trapCoords is a matrix of dimension ntraps x (2 + noccas) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate, and the second column the y-coordinate. The last noccas columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating. studyArea is a 3-column matrix containing the coordinates for the centroids a contiguous grid of cells that define the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column indicates whether the cell is available habitat (=1) or not (=0). All cells must have the same resolution.
centers	N-vector indicating the grid cell (i.e., the row of spatialInputs$studyArea) that contains the true (latent) activity centers for each individual in the population.

Author(s)

Brett T. McClintock

References

Bonner, S. J., and Holmberg J. 2013. Mark-recapture with multiple, non-invasive marks. *Biometrics* 69: 766-775.

McClintock, B. T., Conn, P. B., Alonso, R. S., and Crooks, K. R. 2013. Integrated modeling of bilateral photo-identification data in mark-recapture analyses. *Ecology* 94: 1464-1471.

Royle, J.A., Karanth, K.U., Gopalaswamy, A.M. and Kumar, N.S. 2009. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90: 3233-3244.

See Also

processdataSCR, multimarkClosedSCR, markClosedSCR

Examples

```
#simulate data for data.type="sometimes" using defaults
data<-simdataClosedSCR(data.type="sometimes")
```

---

tiger *Tiger data*

Description

Example tiger data for multimark package.

Format

These spatial capture-recapture data with a single mark type are summarized in a list of length 3 containing the following objects:

Enc.Mat is a 44 x (noccas*ntraps) matrix containing observed encounter histories for 44 tigers across noccas=48 sampling occasions and ntraps=120 traps.

trapCoords is a matrix of dimension ntraps x (2 + noccas) indicating the Cartesian coordinates and operating occasions for the traps, where rows correspond to trap, the first column the x-coordinate, and the second column the y-coordinate. The last noccas columns indicate whether or not the trap was operating on each of the occasions, where '1' indciates the trap was operating and '0' indicates the trap was not operating.

studyArea is a 3-column matrix containing the coordinates for the centroids of the contiguous grid of cells that define the study area and available habitat. Each row corresponds to a grid cell. The first 2 columns indicate the Cartesian x- and y-coordinate for the centroid of each grid cell, and the third column indicates whether the cell is available habitat (=1) or not (=0). The grid cells are 0.336 km^2 resolution.

These data were obtained from the R package SPACECAP and modified by projecting onto a regular rectangular grid consisting of square grid cells (as is required by the spatial capture-recapture models in multimark).

Details

We thank Ullas Karanth, Wildlife Conservation Society, for providing the tiger data for use as an example with this package.

Source

Gopalaswamy, A.M., Royle, J.A., Hines, J.E., Singh, P., Jathanna, D., Kumar, N. and Karanth, K.U. 2012. Program SPACECAP: software for estimating animal density using spatially explicit capture-recapture models. *Methods in Ecology and Evolution* 3:1067-1072.

Royle, J.A., Karanth, K.U., Gopalaswamy, A.M. and Kumar, N.S. 2009. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90: 3233-3244.

See Also

markClosedSCR

Examples

```
data(tiger)
#plot the traps and available habitat within the study area
plotSpatialData(trapCoords=tiger$trapCoords,studyArea=tiger$studyArea)

# This example is excluded from testing to reduce package check time
# Example uses unrealistically low values for nchain, iter, and burnin

# Fit spatial model to tiger data
Enc.Mat<-tiger$Enc.Mat
trapCoords<-tiger$trapCoords
studyArea<-tiger$studyArea
tiger.dot<-markClosedSCR(Enc.Mat,trapCoords,studyArea,iter=100,adapt=50,burnin=50)
summary(tiger.dot$mcmc)
```

Index

∗ **classes**
    multimarkSCRsetup-class, 37
    multimarksetup-class, 39
∗ **data**
    bobcat, 2
    bobcatSCR, 3
    tiger, 62

bobcat, 2, *22*, *27*, *31*, *33*, *50–52*, *55*, *57*, *59*
bobcatSCR, 3

getdensityClosedSCR, 5
getprobsCJS, 6
getprobsClosed, 7
getprobsClosedSCR, 8

make.design.data, *11*, *15*, *19*, *25*, *30*, *36*
markCJS, 9, *15*
markClosed, *12*, 13, *43–45*
markClosedSCR, 16, *46, 47*, *61–63*
mcmc.list, *5–8*, *12*, *15*, *19*, *25*, *30*, *36*, *42*, *44*, *47*
multimarkCJS, *6*, 20, *30*, *40*, *42*, *55, 56*
multimarkClosed, *3*, *7*, *25*, 26, *43–45*, *51*, *57, 58*
multimarkClosedSCR, *4, 5*, *8*, 31, *46, 47*, *53*, *61, 62*
multimarkSCRsetup-class, 37
multimarksetup-class, 39
multimodelCJS, *12*, *25*, 40
multimodelClosed, *15*, *19*, *30, 31*, *36*, 43
multimodelClosedSCR, *19, 20*, 45

plotSpatialData, 48, *60*
processdata, *3*, *12*, *22*, *25*, *28*, *31*, *39*, *42*, *45*, 49, *56*, *58*
processdataSCR, *4*, *34*, *37, 38*, *47*, 51, *62*

simdataCJS, 54
simdataClosed, 56
simdataClosedSCR, 58

tiger, 62