Package 'unmarked'

September 16, 2010

Version 0.8-7

Date 2010-09-15

Type Package

Title Models for Data from Unmarked Animals

Author Ian Fiske, Richard Chandler, Andy Royle

Maintainer Richard Chandler <richard.chandlers@gmail.com>

Depends R (>= 2.9.0), methods, reshape, lattice

Imports graphics, stats, utils

Description Unmarked estimates wildlife parameters for many popular sampling methods including occurrence, point count, and distance data using hierarchical models.

License GPL (>= 3)

LazyLoad yes

LazyData yes

Collate 'classes.R' 'unmarkedEstimate.R' 'mapInfo.R' 'unmarkedFrame.R' 'unmarkedFit.R' 'utils.R' 'getDesign.R' 'colext.R' 'distsamp.R' 'multinomPois.R' 'occu.R' 'occuRN.R' 'pcount.R' 'gmultmix.R' 'unmarkedFitList.R' 'unmarkedLinComb.R'

Encoding UTF-8

URL http://groups.google.com/group/unmarked, http://github.com/ianfiske/unmarked, http://github.com/rbchan/unmarked

Repository CRAN

Date/Publication 2010-09-16 06:37:32

R topics documented:

`unmarked-package` *Models for Data from Unmarked Animals*

Description

Estimate wildlife abundance or occurrence.

Details

Package:	unmarked
Type:	Package
Version:	0.8-7
License:	GPL (>= 3)

Unmarked estimates wildlife parameters for many popular sampling methods including occurrence and point count data.

Overview of Model-fitting Functions: Unmarked provides several functions for fitting integrated likelihood models for wildlife abundance and occurrence to replicated survey data.

`occu` fits occurrence models with no linkage between abundance and detection (MacKenzie et al. 2006).

`occuRN` fits abundance models to presence/absence data by exploiting the link between detection probability and abundance (Royle and Nichols 2003).

`colext` fits the mutli-season occupancy model of MacKenzie et al. 2003.

`pcount` fits N-mixture models for repeated count data (Royle 2004a, Kéry et al 2005).

`distsamp` fits the distsance sampling model of Royle et al. (2004) to distance data recorded in discrete intervals.

`multinomPois` fits the multinomial-Poisson model of Royle 2004b.

`gmultmix` fits a generalized form of the multinomial-mixture model of Royle 2004a that allows for estimating availability and detection probability.

All of these functions allow the user to specify covariates that affect the detection and state processes.

Data: All data is passed to unmarked's estimation functions as a formal S4 class called an unmarkedFrame, which has child classes for each model type. This allows metadata (eg as distance interval cut points, measurement units, etc...) to be stored with the response and covariate data. See `unmarkedFrame` for a detailed description of unmarkedFrames and how to create them.

Model Specification: Most of *unmarked*'s model-fitting functions allow specification of covariates for both the state process and the detection process. Covariates for the detection process (at the site or observation level) and the state process (at the site level) are specified with a double right-hand

sided formula, in that order. Such a formula looks like $x1 + x2 + ... + x3$ $x_1 + x_2 + \ldots + x_n$ where x_1 through x_n are additive covariates of the process of interest. The meaning of these covariates or, what they model, is full described in the help files for the individual functions and is not the same for all functions.

Utility Functions: *unmarked* contains several utility functions for organizing data into the form required by its model-fitting functions. `csvToUMF` converts an appropriately formated comma-separated values (.csv) file to a list containing the components required by model-fitting functions.

Author(s)

Ian Fiske, Richard Chandler, and Andy Royle

References

MacKenzie, D. I. et al. (2006) *Occupancy Estimation and Modeling*. Amsterdam: Academic Press.

Royle, J. A. and Nichols, J. D. (2003) Estimating Abundance from Repeated Presence-Absence Data or Point Counts. *Ecology*, 84(3) pp. 777–790.

Royle, J. A. (2004a) N-Mixture Models for Estimating Population Size from Spatially Replicated Counts. *Biometrics* 60, pp. 108–105.

Royle, J. A., D. K. Dawson, and S. Bates (2004) Modeling abundance effects in distance sampling. *Ecology* 85, pp. 1591–1597.

Kéry, M., Royle, J. A., and Schmid, H. (2005) Modeling Avaian Abundance from Replicated Counts Using Binomial Mixture Models. *Ecological Applications* 15(4), pp. 1450–1461.

Royle, J. A. and Link W. A. (2005) A general class of multinomial mixture models for anuran calling survey data. *Ecology*, **86**(9), pp. 2505–2512.

---

backTransform-methods

Methods for Function backTransform in Package 'unmarked'

Description

Methods for function `backTransform` in Package 'unmarked'. This converts from link-scale to original-scale

Usage

```
## S4 method for signature 'unmarkedFit':
backTransform(obj, type)
## S4 method for signature 'unmarkedEstimate':
backTransform(obj)
```

Arguments

`obj`	Object of appropriate S4 class
`type`	one of names(obj), eg 'state' or 'det'

Methods

obj = "unmarkedEstimate" Typically done internally

obj = "unmarkedFit" Back-transform a parameter from a fitted model. Only possible if no covariates are present. Must specify argument type as one of the values returned by names(obj).

obj = "unmarkedLinComb" Back-transform a predicted value created by `linearComb`. This is done internally by `predict` but can be done explicitly by user.

`birds` *BBS Point Count and Occurrence Data from 2 Bird Species*

Description

Data frames for 2 species from the breeding bird survey (BBS). Each data frame has a row for each site and columns for each sampling event. There is a point count and occurrence–designated by `.bin`– version for each species.

Usage

```
data(birds)
```

Format

`catbird` A data frame of point count observations for the catbird.

`catbird.bin` A data frame of occurrence observations for the catbird.

`woodthrush` A data frame of point count observations for the wood thrush.

`woodthrush.bin` A data frame of point count observations for the wood thrush.

Source

Royle J. N-mixture models for estimating population size from spatially replicated counts. Biometrics. 2004. 60(1):108–115.

Examples

```
data(birds)
```

coef-methods *Methods for Function coef in Package 'unmarked'*

Description

Extract coefficients

Usage

```
## S4 method for signature 'unmarkedFit':
coef(object, type, altNames = TRUE)
## S4 method for signature 'unmarkedEstimate':
coef(object, altNames = TRUE, ...)
## S4 method for signature 'linCombOrBackTrans':
coef(object)
```

Arguments

object	Object of appropriate S4 class
type	Either 'state' or 'det'
altNames	Return specific names for parameter estimates?
...	Further arguments. Not currently used

Value

A named numeric vector of parameter estimates.

Methods

object = "linCombOrBackTrans" Object from linearComb

object = "unmarkedEstimate" unmarkedEstimate object

object = "unmarkedFit" Fitted model

colext *Fit the colonization-extinction model.*

Description

Estimate parameters of the colonization-extinction model, including covariate-dependent rates and detection process.

Usage

```
colext(psiformula= ~1, gammaformula =  ~ 1, epsilonformula = ~ 1,
pformula = ~ 1, data, starts, method="BFGS", control=list(),
    se=TRUE)
```

Arguments

psiformula	Right-hand sided formula for the initial probability of occupancy at each site.
gammaformula	Right-hand sided formula for colonization probability.
epsilonformula	Right-hand sided formula for extinction probability.
pformula	Right-hand sided formula for detection probability.
data	unmarkedMultFrame object that supplies the data (see unmarkedMultFrame).
starts	optionally, initial values for parameters in the optimization.
method	Optimization method used by optim.
control	Other arguments passed to optim.
se	logical specifying whether or not to compute standard errors.

Details

This function fits the colonization-extinction model of MacKenzie et al (2003). The colonization and extinction rates can be modeled with covariates that vary yearly at each site using a logit link. These covariates are supplied by special unmarkedMultFrame yearlySiteCovs slot. These parameters are specified using the gammaformula and epsilonformula arguments. The initial probability of occupancy is modeled by covariates specified in the psiformula.

The conditional detection rate can also be modeled as a function of covariates that vary at the secondary sampling period (ie., repeat visits). These covariates are specified by the first part of the formula argument and the data is supplied via the usual obsCovs slot.

The projected and smoothed trajectories (Weir et al 2009) can be obtained from the smoothed.mean and projected.mean slots (see examples).

Value

unmarkedFitColExt object describing model fit.

References

MacKenzie, D.I. et al. (2002) Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology, 83(8), 2248-2255.

MacKenzie, D. I. et al. (2006) *Occupancy Estimation and Modeling*.Amsterdam: Academic Press.

Weir L. A., Fiske I. J., Royle J. (2009) Trends in Anuran Occupancy from Northeastern States of the North American Amphibian Monitoring Program. Herpetological Conservation and Biology. 4(3):389-402.

See Also

nonparboot, unmarkedMultFrame, and formatMult

Examples

```
data(frogs)
umf <- formatMult(masspcru)
obsCovs(umf) <- scale(obsCovs(umf))

## Use 1/4 of data just for run speed in example
umf <- umf[which((1:numSites(umf)) %% 4 == 0),]

## constant transition rates
(fm <- colext(psiformula = ~ 1,
gammaformula = ~ 1,
epsilonformula = ~ 1,
pformula = ~ JulianDate + I(JulianDate^2), umf, control = list(trace=1, maxit=1e4)))

## get the trajectory estimates
smoothed(fm)
projected(fm)

## Not run:
## Find bootstrap standard errors for smoothed trajectory
fm <- nonparboot(fm, B = 100)  # This takes a while!
fm@smoothed.mean.bsse

## End(Not run)

## Not run:
## try yearly transition rates
yearlySiteCovs(umf) <- data.frame(year = factor(rep(1:7, numSites(umf))))
(fm.yearly <- colext(psiformula = ~ 1,
gammaformula = ~ year,
epsilonformula = ~ year,
pformula = ~ JulianDate + I(JulianDate^2), umf,
control = list(trace=1, maxit=1e4)))

## End(Not run)
```

`confint-methods` *Methods for Function confint in Package 'unmarked'*

Description

Methods for function `confint` in Package 'unmarked'

Usage

```
## S4 method for signature 'unmarkedBackTrans':
confint(object, parm, level)
## S4 method for signature 'unmarkedEstimate':
```

```
confint(object, parm, level)
## S4 method for signature 'unmarkedLinComb':
confint(object, parm, level)
## S4 method for signature 'unmarkedFit':
confint(object, parm, level, type, method)
```

Arguments

object	Object of appropriate S4 class
parm	Name of parameter(s) of interest
level	Level of confidence
type	Either "state" or "det"
method	Either "normal" or "profile"

Value

A vector of lower and upper confidence intervals. These are asymtotic unless method='profile' is used on unmarkedFit objects in which case they are profile likelihood intervals.

See Also

unmarkedFit-class

---

csvToUMF *Convert .CSV File to an unmarkedFrame*

Description

This function converts an appropriatedly formated comma-separated values file (.csv) to a format usable by *unmarked*'s fitting functions (see *Details*).

Usage

```
csvToUMF(filename, long=FALSE, type, species, ...)
```

Arguments

filename	string describing filename of file to read in
long	FALSE if file is in long format or TRUE if file is in long format (see *Details*)
species	if data is in long format with multiple species, then this can specify a particular species to extract if there is a column named "species".
type	specific type of unmarkedFrame.
...	further arguments to be passed to the unmarkedFrame constructor.

Details

This function provides a quick way to take a .csv file with headers named as described below and provides the data required and returns of data in the format required by the model-fitting functions in unmarked. The .csv file can be in one of 2 formats: long or wide. See the first 2 lines of the *examples* for what these formats look like.

The .csv file is formatted as follows:

- col 1 is site labels.
- if data is in long format, col 2 is date of observation.
- next J columns are the observations (y) - counts or 0/1's.
- next is a series of columns for the site variables (one column per variable). The column header is the variable name.
- next is a series of columns for the observation-level variables. These are in sets of J columns for each variable, e.g., var1-1 var1-2 var1-3 var2-1 var2-2 var2-3, etc. The column header of the first variable in each group must indicate the variable name.

Value

an unmarkedFrame object

Author(s)

Ian Fiske <ianfiske@gmail.com>

Examples

```
# examine a correctly formatted long .csv
head(read.csv(system.file("csv","frog2001pcru.csv", package="unmarked")))

# examine a correctly formatted wide .csv
head(read.csv(system.file("csv","widewt.csv", package="unmarked")))

# convert them!
dat1 <- csvToUMF(system.file("csv","frog2001pcru.csv", package="unmarked"),
                 long = TRUE, type = "unmarkedFrameOccu")
dat2 <- csvToUMF(system.file("csv","frog2001pfer.csv", package="unmarked"),
                 long = TRUE, type = "unmarkedFrameOccu")
dat3 <- csvToUMF(system.file("csv","widewt.csv", package="unmarked"),
                 long = FALSE, type = "unmarkedFrameOccu")
```

`detFuns` *Distance-sampling detection functions and associated density functions*

Description

These functions represent the currently available detection functions used for modeling line and point transect data with distsamp. Detection functions begin with "g", and density functions begin with a "d".

Usage

```
gxhn(x, sigma)
gxexp(x, rate)
gxhaz(x, shape, scale)

dxhn(x, sigma)
dxexp(x, rate)
dxhaz(x, shape, scale)
drhn(r, sigma)
drexp(r, rate)
drhaz(r, shape, scale)
```

Arguments

x	Perpendicular distance
r	Radial distance
sigma	Shape parameter of half-normal detection function
rate	Shape parameter of negative-exponential detection function
shape	Shape parameter of hazard-rate detection function
scale	Scale parameter of hazard-rate detection function

See Also

distsamp for example of using these for plotting detection function

Examples

```
# Detection probabilities at 25m for range of half-normal sigma values.
round(gxhn(25, 10:15), 2)

# Plot negative exponential distributions
plot(function(x) gxexp(x, rate=10), 0, 50, xlab="distance",
    ylab="Detection probability")
plot(function(x) gxexp(x, rate=20), 0, 50, add=TRUE, lty=2)
plot(function(x) gxexp(x, rate=30), 0, 50, add=TRUE, lty=3)

# Plot half-normal probability density functions for line- and point-transects
par(mfrow=c(2, 1))
plot(function(x) dxhn(x, 20), 0, 50, xlab="distance",
    ylab="Probability density", main="Line-transect")
plot(function(x) drhn(x, 20), 0, 50, xlab="distance",
    ylab="Probability density", main="Point-transect")
```

`distsamp` *Fit the hierarchical distance sampling model of Royle et al. (2004)*

Description

Fit the hierarchical distance sampling model of Royle et al. (2004) to line or point transect data recorded in discrete distance intervals.

Usage

```
distsamp(formula, data, keyfun=c("halfnorm", "exp",
    "hazard", "uniform"), output=c("density", "abund"),
    unitsOut=c("ha", "kmsq"), starts, method="BFGS",
    control=list(), se=TRUE)
```

Arguments

`formula`	Double right-hand formula describing detection covariates followed by abundance covariates. ~1 ~1 would be a null model.
`data`	object of class `unmarkedFrameDS`, containing response matrix, covariates, distance interval cut points, survey type ("line" or "point"), transect lengths (for survey = "line"), and units ("m" or "km") for cut points and transect lengths. See example for set up.
`keyfun`	One of the following detection functions: "halfnorm", "hazard", "exp", or "uniform." See details.
`output`	Model either "density" or "abund"
`unitsOut`	Units of density. Either "ha" or "kmsq" for hectares and square kilometers, respectively.
`starts`	Vector of starting values for parameters.
`method`	Optimization method used by `optim`.
`control`	Other arguments passed to `optim`.
`se`	logical specifying whether or not to compute standard errors.

Details

Unlike conventional distance sampling, which uses the 'conditional on detection' likelihood formulation, this model is based upon the unconditional likelihood and thus allows for modeling both abundance and detection function parameters.

The latent transect-level abundance distribution $f(N|\theta)$ is currently assumed to be Poisson with mean λ.

The detection process is modeled as multinomial: $y_{ij} \sim Multinomial(N_i, pi_{ij})$, where pi_ij is the multinomial cell probability for transect i in distance class j. These are computed based upon a detection function $g(x|\sigma)$, such as the half-normal, negative exponential, or hazard rate.

Parameters λ and σ can be vectors affected by transect-specific covariates using the log link.

Value

unmarkedFitDS object (child class of unmarkedFit-class) describing the model fit.

Note

You cannot use obsCovs.

Author(s)

Richard Chandler <richard.chandlers@gmail.com>

References

Royle, J. A., D. K. Dawson, and S. Bates (2004) Modeling abundance effects in distance sampling. *Ecology* 85, pp. 1591-1597.

See Also

unmarkedFit-class fitList, formatDistData, parboot, sight2perpdist, detFuns. Also look at vignette("distsamp").

Examples

```
## Line transect examples

data(linetran)

ltUMF <- with(linetran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
siteCovs = data.frame(Length, area, habitat),
dist.breaks = c(0, 5, 10, 15, 20),
tlength = linetran$Length * 1000, survey = "line", unitsIn = "m")
})

ltUMF
summary(ltUMF)
hist(ltUMF)

# Half-normal detection function. Density output (log scale). No covariates.
(fm1 <- distsamp(~ 1 ~ 1, ltUMF))

# Some methods to use on fitted model
summary(fm1)
backTransform(fm1, type="state")       # animals / ha
exp(coef(fm1, type="state", altNames=TRUE))     # same
backTransform(fm1, type="det")         # half-normal SD
hist(fm1, xlab="Distance (m)") # Only works when there are no detection covars

# Halfnormal. Covariates affecting both density and and detection.
(fm2 <- distsamp(~area + habitat ~ habitat, ltUMF))
```

```
# Hazard-rate detection function.
(fm3 <- distsamp(~ 1 ~ 1, ltUMF, keyfun="hazard"))

# Plot detection function.
fmhz.shape <- exp(coef(fm3, type="det"))
fmhz.scale <- exp(coef(fm3, type="scale"))
plot(function(x) gxhaz(x, shape=fmhz.shape, scale=fmhz.scale), 0, 25,
xlab="Distance (m)", ylab="Detection probability")

## Point transect example

## Not run:
data(pointtran)

ptUMF <- with(pointtran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4, dc5),
siteCovs = data.frame(area, habitat),
dist.breaks = seq(0, 25, by=5), survey = "point", unitsIn = "m")
})

# Half-normal.
(fmp1 <- distsamp(~ 1 ~ 1, ptUMF))
hist(fmp1, ylim=c(0, 0.07), xlab="Distance (m)")

## End(Not run)
```

`fitList` *constructor of unmarkedFitList objects*

Description

Organize models for model selection or model-averaged prediction.

Usage

```
fitList(..., fits)
```

Arguments

`...`	Fitted models. Preferrably named.
`fits`	An alternative way of providing the models. A (preferrably named) list of fitted models.

Note

Two requirements exist to conduct AIC-based model-selection and model-averaging in unmarked. First, the data objects (ie, unmarkedFrames) must be identical among fitted models. Second, the response matrix must be identical among fitted models after missing values have been removed. This means that if a response value was removed in one model due to missingness, it needs to be removed from all models.

Author(s)

Richard Chandler <rchandler@nrc.umass.edu>

Examples

```
# Fit some N-mixture models
data(linetran)
(dbreaksLine <- c(0, 5, 10, 15, 20))
lengths <- linetran$Length * 1000

ltUMF <- with(linetran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
siteCovs = data.frame(Length, area, habitat), dist.breaks = dbreaksLine,
tlength = lengths, survey = "line", unitsIn = "m")
})

fm1 <- distsamp(~ 1 ~1, ltUMF)
fm2 <- distsamp(~ area ~1, ltUMF)
fm3 <- distsamp( ~ 1 ~area, ltUMF)

## Two methods of creating an unmarkedFitList using fitList()

# Method 1
fmList <- fitList(Null=fm1, .area=fm2, area.=fm3)

# Method 2. Note that the arugment name "fits" must be included in call.
models <- list(Null=fm1, .area=fm2, area.=fm3)
fmList <- fitList(fits = models)


# Model-averaged prediction
predict(fmList, type="state")

# Model selection
modSel(fmList, nullmod="Null")
```

---

fitted-methods *Methods for Function fitted in Package 'unmarked'*

Description

Extracted fitted values from a fitted model.

Usage

```
## S4 method for signature 'unmarkedFit':
fitted(object, na.rm = FALSE)
## S4 method for signature 'unmarkedFitColExt':
fitted(object, na.rm = FALSE)
```

```
## S4 method for signature 'unmarkedFitOccu':
fitted(object, na.rm = FALSE)
## S4 method for signature 'unmarkedFitOccuRN':
fitted(object, K, na.rm = FALSE)
## S4 method for signature 'unmarkedFitPCount':
fitted(object, K, na.rm = FALSE)
## S4 method for signature 'unmarkedFitDS':
fitted(object, na.rm = FALSE)
```

Arguments

object	A fitted model of appropriate S4 class
K	Integer specifying upper bound of integration.
na.rm	Logical. Should missing values be removed from data?

Value

Returns a matrix of expected values

Methods

object = "unmarkedFit" A fitted model
object = "unmarkedFitColExt" A model fit by colext
object = "unmarkedFitOccu" A model fit by occu
object = "unmarkedFitOccuRN" A model fit by occuRN
object = "unmarkedFitPCount" A model fit by pcount
object = "unmarkedFitDS" A model fit by distsamp

---

formatDistData *Convert individual-level distance data to the transect-level format required by distsamp*

Description

Convert individual-level distance data to the transect-level format required by distsamp

Usage

```
formatDistData(distData, distCol, transectNameCol, dist.breaks)
```

Arguments

distData	data.frame where each row is a detected individual. Must have at least 2 columns. One for distances and the other for transect names.
distCol	character, the column name containing distances
transectNameCol	character, column name containing transect names
dist.breaks	numeric vector of distance interval cutpoints. Length must equal J+1.

Details

This function creates a site (M) by distance interval (J) response matrix from a data.frame containing the detection distances for each individual and the transect names.

Value

An M x J data.frame containing the tabulated detections in each distance interval for each transect. Transect names will become rownames and colnames will be y.1, y.2, ..., y.J.

Note

It is very important that the factor containing transect names contains levels for all the transects surveyed. This includes those where no individuals were detected. See the example for how to add levels to a factor.

See Also

distsamp, unmarkedFrame

Examples

```
# Create a data.frame containing distances of animals detected
# along 4 transects.
dat <- data.frame(transect=gl(4,5, labels=letters[1:4]), distance=rpois(20, 10))
dat

# Look at your transect names.
levels(dat$transect)

# Suppose that you also surveyed a transect named "e" where no animals were
# detected. You must add it to the levels of dat$transect
levels(dat$transect) <- c(levels(dat$transect), "e")
levels(dat$transect)

# Distance cut points defining distance intervals
cp <- c(6, 8, 10, 12, 14, 18)

# Create formated response data.frame
yDat <- formatDistData(dat, "distance", "transect", cp)
yDat

# Now you could merge yDat with transect-level covariates and
# then use unmarkedFrameDS to prepare data for distsamp
```

formatMult *Create unmarkedMultFrame from Long Format Data Frame*

Description

This convenience function converts multi-year data in long format to unmarkedMultFrame Object. See Details for more information.

Usage

```
formatMult(df.in)
```

Arguments

df.in	a data.frame appropriately formatted (see Details).

Details

df.in is a data frame with columns formatted as follows:

Column 1 = year number
Column 2 = site name or number
Column 3 = julian date or chronological sample number during year
Column 4 = observations (y)
Column 5 – Final Column = covariates

Note that if the data is already in wide format, it may be easier to create an unmarkedMultFrame object directly with a call to unmarkedMultFrame.

Value

unmarkedMultFrame object

formatWideLong *Convert between wide and long data formats.*

Description

Convert a data.frame between wide and long formats.

Usage

```
formatWide(dfin, sep = ".", obsToY, type, ...)
formatLong(dfin, species = NULL, type)
```

Arguments

dfin	A data.frame to be reformatted.
sep	A seperator of column names in wide format.
obsToY	Optional matrix specifying relationship between covariate column structure and response matrix structure.
type	Type of unmarkedFrame to create?
species	Character name of species response column
...	Further arguments

Details

In order for these functions to work, the columns of dfin need to be in the correct order. formatLong requires that the columns are in the following scheme:

1. site name or number.
2. date or observation number.
3. response variable (detections, counts, etc).
4. The remaining columns are observation-level covariates.

formatWide requires particular names for the columns. The column order for formatWide is

1. (optional) site name, named "site".
2. response, named "y.1", "y.2", ..., "y.J".
3. columns of site-level covariates, each with a relevant name per column.
4. groups of columns of observation-level covariates, each group having the name form "someObsCov.1", "someObsCov.2", ..., "someObsCov.J".

Value

A data.frame

See Also

csvToUMF

`frogs` *2001 Delaware North American Amphibian Monitoring Program Data*

Description

frogs contains NAAMP data for Pseudacris feriarum (pfer) and Pseudacris crucifer (pcru) in 2001.

Usage

```
data(frogs)
```

Format

pcru.y matrix of observed calling indices for pcru

pcru.bin matrix of detections for pcru

pcru.data array of covariates measured at the observation-level for pcru

pfer.y matrix of observed calling indices for pfer

pfer.bin matrix of detections for pfer

pfer.data array of covariates measured at the observation-level for pfer

Details

The rows of pcru.y, pcru.bin, pfer.y, and pfer.bin correspond to sites and columns correspond to visits to each site. The first 2 dimensions of pfer.data and pcru.data are matrices of covariates that correspond to the observation matrices (sites $\times$ observation), with the 3rd dimension corresponding to separate covariates.

Source

https://www.pwrc.usgs.gov/naamp/

References

Mossman MJ, Weir LA. North American Amphibian Monitoring Program (NAAMP). Amphibian Declines: the conservation status of United States species. University of California Press, Berkeley, California, USA. 2005:307-313.

Examples

```
data(frogs)
str(pcru.data)
```

getP-methods *Methods for Function getP in Package 'unmarked'*

Description

Methods for function `getP` in Package 'unmarked'. These methods return a matrix of detection probabilities.

Methods

object = "unmarkedFit" A fitted model object

object = "unmarkedFitDS" A fitted model object

object = "unmarkedFitMPois" A fitted model object

object = "unmarkedFitGMM" A fitted model object

gf *Green frog count index data*

Description

Multinomial calling index data.

Usage

```
data(gf)
```

Format

A list with 2 components

gf.data 220 x 3 matrix of count indices

gf.obs list of covariates

References

Royle, J. Andrew, and William A. Link. 2005. A General Class of Multinomial Mixture Models for Anuran Calling Survey Data. Ecology 86, no. 9: 2505–2512.

Examples

```
data(gf)
str(gf.data)
str(gf.obs)
```

gmultmix *Generalized multinomial-mixture model*

Description

A three level hierarchical model for designs involving primary and secondary sampling periods. Virtually any sampling method that results in multinomial outcomes can be used during the secondary sampling period. Examples include removal sampling and double observer sampling. The three model parameters are abundance, availability, and detection probability.

Usage

```
gmultmix(lambdaformula, phiformula, pformula, data, mixture = c("P", "NB"), K,
    starts, method = "BFGS", control = list(), se = TRUE)
```

Arguments

lambdaformula	Righthand side (RHS) formula describing abundance covariates
phiformula	RHS formula describing availability covariates
pformula	RHS formula describing detection covariates
data	An object of class unmarkedFrameGMM
mixture	Either "P" or "NB" for Poisson and Negative Binomial mixing distributions.
K	The upper bound of integration
starts	Starting values
method	Optimization method used by optim
control	List of control arguments passed to optim
se	Logical. Should standard errors be calculated?

Details

The latent transect-level super-population abundance distribution $f(M|\theta)$ can be set as either a Poisson or a negative binomial random variable, depending on the setting of the mixture argument. mixture = "P" or mixture = "NB" select the Poisson or negative binomial distribution respectively. The mean of M_i is λ_i. If $M_i \sim NB$, then an additional parameter, α, describes dispersion (lower α implies higher variance).

The number of individuals available for detection at time t is a modeled as binomial: $N_{it} \sim Binomial(M_i, \phi_{\mathbf{i}}\mathbf{j})$.

The detection process is modeled as multinomial: $y_{ij} \sim Multinomial(N_i t, pi_{itj})$, where $pi_i tj$ is the multinomial cell probability for plot i at time t on occasion j.

Cell probabilities are computed via a user-defined function related to the sampling design. Alternatively, the default functions removalPiFun or doublePiFun can be used for equal-interval removal sampling or double observer sampling. Note that the function for computing cell probabilites is specified when setting up the data using unmarkedFrameGMM.

Parameters λ, ϕ and p can be modeled as linear functions of covariates using the log, logit and logit links respectively.

Value

An object of class unmarkedFitGMM.

Note

Three types of covariates can be supplied, site-level, site-by-year-level, and observation-level. These must be formatted correctly when organizing the data with `unmarkedFrameGMM`

Author(s)

Richard Chandler <richard.chandlers@gmail.com> and Andy Royle

References

Royle, J. A. (2004) Generalized estimators of avian abundance from count survey data. *Animal Biodiversity and Conservation* 27, pp. 375–386.

See Also

`unmarkedFrameGMM` for setting up the data and metadata. `multinomPois` for surveys where no secondary sampling periods were used. Example functions to calculate multinomial cell probabilities are described `piFuns`

Examples

```
# Simulate data using the multinomial-Poisson model with a
# repeated constant-interval removal design.

n <- 100  # number of sites
T <- 4    # number of primary periods
J <- 3    # number of secondary periods

lam <- 3
phi <- 0.5
p <- 0.3

#set.seed(26)
y <- array(NA, c(n, T, J))
M <- rpois(n, lam)          # Local population size
N <- matrix(NA, n, T)       # Individuals available for detection

for(i in 1:n) {
    N[i,] <- rbinom(T, M[i], phi)
    y[i,,1] <- rbinom(T, N[i,], p)    # Observe some
    Nleft1 <- N[i,] - y[i,,1]         # Remove them
    y[i,,2] <- rbinom(T, Nleft1, p)   # ...
    Nleft2 <- Nleft1 - y[i,,2]
```

```
    y[i,,3] <- rbinom(T, Nleft2, p)
    }

y.ijt <- cbind(y[,1,], y[,2,], y[,3,], y[,4,])


umf1 <- unmarkedFrameGMM(y=y.ijt, numPrimary=T, type="removal")

(m1 <- gmultmix(~1, ~1, ~1, data=umf1))

backTransform(m1, type="lambda")        # Individuals per plot
backTransform(m1, type="phi")           # Probability of being avilable
(p <- backTransform(m1, type="det"))    # Probability of detection
p <- coef(p)

# Multinomial cell probabilities under removal design
c(p, (1-p) * p, (1-p)^2 * p)

# Or more generally:
head(getP(m1))
```

`imputeMissing` *A function to impute missing entries in continuous obsCovs*

Description

This function uses an ad-hoc averaging approach to impute missing entries in obsCovs. The missing entry is replaced by an average of the average for the site and the average for the visit number.

Usage

```
imputeMissing(umf, whichCovs = seq(length=ncol(obsCovs(umf))))
```

Arguments

`umf`	The data set who's obsCovs are being imputed.
`whichCovs`	An integer vector giving the indices of the covariates to be imputed. This defaults to all covariates in `obsCovs`.

Value

A version of `umf` that has the requested obsCovs imputed.

Author(s)

Ian Fiske

Examples

```
data(frogs)
pcru.obscovs <- data.frame(MinAfterSunset=as.vector(t(pcru.data[,,1])),
     Wind=as.vector(t(pcru.data[,,2])),
     Sky=as.vector(t(pcru.data[,,3])),
     Temperature=as.vector(t(pcru.data[,,4])))
pcruUMF <- unmarkedFrameOccu(y = pcru.bin, obsCovs = pcru.obscovs)
pcruUMF.i1 <- imputeMissing(pcruUMF)
pcruUMF.i2 <- imputeMissing(pcruUMF, whichCovs = 2)
```

---

lambda2psi *Convert Poisson mean (lambda) to probability of occurrence (psi).*

Description

Abundance and occurrence are fundamentally related.

Usage

```
lambda2psi(lambda)
```

Arguments

lambda	Numeric vector with values >= 0

Value

A vector of psi values of the same length as lambda.

See Also

pcount, multinomPois, distsamp

Examples

```
lambda2psi(0:5)
```

linearComb-methods *Methods for Function linearComb in Package 'unmarked'*

Description

Methods for function `linearComb` in Package 'unmarked'

Methods

obj = "unmarkedEstimate", coefficients = "matrixOrVector" Typically called internally

obj = "unmarkedFit", coefficients = "matrixOrVector" Returns linear combinations of parameters from a fitted model. Coefficients are supplied through coefficients. The required argument `type` specifies which model estimate to use. You can use `names(fittedmodel)` to view possible values for the `type` argument.

Examples

```
data(ovendata)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])), type = "removal")
fm <- multinomPois(~ 1 ~ ufp + trba, ovenFrame)
linearComb(fm, c(1, 0.5, 0.5), type = "state")
linearComb(fm, matrix(c(1, 0.5, 0.5, 1, 0, 0, 1, 0, 0.5), 3, 3,
  byrow=TRUE), type="state")
```

linetran *Simulated line transect data*

Description

Response matrix of animals detected in four distance classes plus transect lengths and two covariates.

Usage

```
data(linetran)
```

Format

A data frame with 12 observations on the following 7 variables.

`dc1` Counts in distance class 1 [0-5 m)

`dc2` Counts in distance class 2 [5-10 m)

`dc3` Counts in distance class 3 [10-15 m)

`dc4` Counts in distance class 4 [15-20 m)

`Length` Transect lengths in km

`area` Numeric covariate

`habitat` a factor with levels `A` and `B`

Examples

```
data(linetran)
linetran

# Format for distsamp()
ltUMF <- with(linetran, {
        unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
        siteCovs = data.frame(Length, area, habitat),
        dist.breaks = c(0, 5, 10, 15, 20),
        tlength = linetran$Length * 1000, survey = "line", unitsIn = "m")
        })
```

---

mallard *Mallard count data*

Description

Mallard repeated count data and covariates

Usage

```
data(mallard)
```

Format

A list with 3 components

mallard.y response matrix

mallard.site site-specific covariates

mallard.obs survey-specific covariates

References

Kéry, M., Royle, J. A., and Schmid, H. (2005) Modeling Avaian Abundance from Replicated Counts Using Binomial Mixture Models. *Ecological Applications* 15(4), pp. 1450–1461.

Examples

```
data(mallard)
str(mallard.y)
str(mallard.site)
str(mallard.obs)
```

masspcru *Massachusetts North American Amphibian Monitoring Program Data*

Description

masspcru contains NAAMP data for Pseudacris crucifer (pcru) in Massachusetts from 2001 to 2007 in the raw long format.

Usage

```
data(frogs)
```

Format

Data frame with

SurveyYear Year of data collection.

RouteNumStopNum Stop number.

JulianDate Day of year.

Pcru Observed calling index.

MinAfterSunset Minutes after sunset of the observation.

Temperature Temperature measured during observation.

Details

These data come from the North American Amphibian Monitoring Program. Please see the reference below for more details.

Source

https://www.pwrc.usgs.gov/naamp/

References

Mossman MJ, Weir LA. North American Amphibian Monitoring Program (NAAMP). Amphibian Declines: the conservation status of United States species. University of California Press, Berkeley, California, USA. 2005:307-313.

Examples

```
data(masspcru)
str(masspcru)
```

modSel *Model selection results from an unmarkedFitList*

Description

Model selection results from an unmarkedFitList

Arguments

object	an object of class "unmarkedFitList" created by the function fitList.
nullmod	optional character naming which model in the fitList contains results from the null model. Only used in calculation of Nagelkerke's R-squared index.

Value

A S4 object with the following slots

Full	data.frame with formula, estimates, standard errors and model selection information. Converge is optim convergence code. CondNum is model condition number. n is the number of sites. delta is delta AIC. cumltvWt is cumulative AIC weight. Rsq is Nagelkerke's (1991) R-squared index, which is only returned when the nullmod argument is specified.
Names	matrix referencing column names of estimates (row 1) and standard errors (row 2).

Note

Two requirements exist to conduct AIC-based model-selection and model-averaging in unmarked. First, the data objects (ie, unmarkedFrames) must be identical among fitted models. Second, the response matrix must be identical among fitted models after missing values have been removed. This means that if a response value was removed in one model due to missingness, it needs to be removed from all models.

Author(s)

Richard Chandler <rchandler@nrc.umass.edu>

References

Nagelkerke, N.J.D. (2004) A Note on a General Definition of the Coefficient of Determination. *Biometrika* 78, pp. 691-692.

Examples

```
data(linetran)
(dbreaksLine <- c(0, 5, 10, 15, 20))
lengths <- linetran$Length * 1000

ltUMF <- with(linetran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
siteCovs = data.frame(Length, area, habitat), dist.breaks = dbreaksLine,
tlength = lengths, survey = "line", unitsIn = "m")
})

fm1 <- distsamp(~ 1 ~1, ltUMF)
fm2 <- distsamp(~ area ~1, ltUMF)
fm3 <- distsamp( ~ 1 ~area, ltUMF)

fl <- fitList(Null=fm1, A.=fm2, .A=fm3)
fl

ms <- modSel(fl, nullmod="Null")

ms
summary(ms)

coef(ms)                              # Estimates only
SE(ms)                                # Standard errors only
(toExport <- as(ms, "data.frame"))    # Estimates, SEs, and AIC.
```

---

multinomPois *Multinomial-Poisson Mixtures*

Description

Fit the multinomial-Poisson abundance mixture model.

Usage

```
multinomPois(formula, data, starts, method = "BFGS", control = list(),
se = TRUE)
```

Arguments

formula	double right-hand side formula for detection and abundance covariates, in that order.
data	unmarkedFrame supplying data.
starts	vector of starting values.
method	Optimization method used by optim.
control	Other arguments passed to optim.
se	logical specifying whether or not to compute standard errors.

Details

This function takes advantage of the closed form of the integrated likelihood when a latent Poisson distribution is assumed for abundance at each site and a multinomial distribution is taken for the observation state. Many common sampling methods can be framed in this context. For example, double-observer point counts, removal sampling, and distance sampling can all be analyzed with this function by specifying the proper multinomial cell probablilities. This is done with by supplying the appropriate function (piFun) argument. removalPiFun and doublePiFun are supplied as example cell probability functions.

Value

unmarkedFit object describing the model fit.

Author(s)

Ian Fiske

References

Royle, J. A., Dawson, D., & Bates, S. (2004). Modeling abundance effects in distance sampling. Ecology, 85(6), 1591-1597.

Royle, J. A. (2004). Generalized estimators of avian abundance from count survey data. Animal Biodiversity and Conservation, 27(1), 375-386.

Royle, J. A., & Dorazio, R. M. (2006). Hierarchical Models of Animal Abundance and Occurrence. Journal Of Agricultural Biological And Environmental Statistics, 11(3), 249.

See Also

piFuns

Examples

```
data(ovendata)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])), type = "removal")
(fm1 <- multinomPois(~ 1 ~ ufp + trba, ovenFrame))
(fm2 <- multinomPois(~ 1 ~ ufp, ovenFrame))
(fm3 <- multinomPois(~ 1 ~ trba, ovenFrame))
(fm4 <- multinomPois(~ 1 ~ 1, ovenFrame))
fmList <- fitList(Global=fm1, ufp=fm2, trba=fm3, Null=fm4)

# Model selection
modSel(fmList, nullmod="Null")
```

nonparboot-methods *Nonparametric bootstrapping in unmarked*

Description

Call nonparboot on an unmarkedFit to obtain non-parametric bootstrap samples. These can then be used by vcov in order to get bootstrap estimates of standard errors.

Details

Calling nonparboot on an unmarkedFit returns the original unmarkedFit, with the bootstrap samples added on. Then subsequent calls to vcov with the argument method="nonparboot" will use these bootstrap samples. Additionally, standard errors of derived estimates from either linearComb or backTransform can be instructed to use bootstrap samples by providing the argument method = "nonparboot".

Methods

signature(object = "unmarkedFit") Obtain nonparametric bootstrap samples for a general unmarkedFit.

signature(object = "unmarkedFitColExt") Obtain nonparametric bootstrap samples for colext fits.

signature(object = "unmarkedFitDS") Obtain nonparametric bootstrap samples for a distsamp fits.

signature(object = "unmarkedFitMPois") Obtain nonparametric bootstrap samples for a distsamp fits.

signature(object = "unmarkedFitOccu") Obtain nonparametric bootstrap samples for a occu fits.

signature(object = "unmarkedFitOccuRN") Obtain nonparametric bootstrap samples for a occuRN fits.

signature(object = "unmarkedFitPCount") Obtain nonparametric bootstrap samples for a pcount fits.

Examples

```
data(ovendata)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])), type = "removal")
(fm <- multinomPois(~ 1 ~ ufp + trba, ovenFrame))
fm <- nonparboot(fm, B = 20) # should use larger B in real life.
vcov(fm, method = "hessian")
vcov(fm, method = "nonparboot")
avg.abundance <- backTransform(linearComb(fm, type = "state", coefficients = c(1, 0, 0)))

## Bootstrap sample information propagates through to derived quantities.
vcov(avg.abundance, method = "hessian")
vcov(avg.abundance, method = "nonparboot")
SE(avg.abundance, method = "nonparboot")
```

occu *Fit the MacKenzie Occupancy Model*

Description

This function estimates the standard occupancy model of MacKenzie et al (2002).

Usage

```
occu(formula, data, knownOcc=numeric(0), starts, method="BFGS",
    control=list(), se=TRUE)
```

Arguments

formula	double right-hand side formula describing covariates of detection and occupancy in that order.
data	an unmarkedFrameOccu object (see unmarkedFrame)..
knownOcc	vector of sites that are known to be occupied.
starts	vector of parameter starting values.
method	Optimization method used by optim.
control	Other arguments passed to optim.
se	logical specifying whether or not to compute standard errors.

Details

See unmarkedFrame for a description of how to supply data to the umf argument. occu fits the standard occupancy model based on zero-inflated binomial models (MacKenzie et al. 2006, Royle and Dorazio 2008). The occupancy state process (z_i) of site i is modeled as

$$z_i \sim Bernoulli(\psi_i)$$

The observation process is modeled as

$$y_{ij}|z_i \sim Bernoulli(z_i p_{ij})$$

Covariates of ψ_i and p_{ij} are modeled using the logit link according to the formula argument. The formula is a double right-hand sided formula like ~ detform ~ occform where detform is a formula for the detection process and occform is a formula for the partially observed occupancy state. See formula for details on constructing model formulae in R.

Value

unmarkedFitOccu object describing the model fit.

Author(s)

Ian Fiske

References

MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology 83, no. 8 (2002): 2248-2255.

MacKenzie, D. I. et al. (2006) *Occupancy Estimation and Modeling*. Amsterdam: Academic Press. Royle, J. A. and R. Dorazio. (2008).

Examples

```
data(frogs)
pferUMF <- unmarkedFrameOccu(pfer.bin)
plot(pferUMF, panels=4)
# add some fake covariates for illustration
siteCovs(pferUMF) <- data.frame(sitevar1 = rnorm(numSites(pferUMF)))

# observation covariates are in site-major, observation-minor order
obsCovs(pferUMF) <- data.frame(obsvar1 = rnorm(numSites(pferUMF) * obsNum(pferUMF)))

(fm <- occu(~ obsvar1 ~ 1, pferUMF))

confint(fm, type='det', method = 'normal')
confint(fm, type='det', method = 'profile')

# estimate detection effect at obsvars=0.5
(lc <- linearComb(fm['det'],c(1,0.5)))

# transform this to probability (0 to 1) scale and get confidence limits
(btlc <- backTransform(lc))
confint(btlc, level = 0.9)
```

---

occuRN *Fit the Occupancy model of Royle and Nichols...*

Description

Fit the Occupancy model of Royle and Nichols

Usage

```
occuRN(formula, data, K=25, starts, method="BFGS", control=list(),
    se=TRUE)
```

Arguments

formula	double right-hand side formula describing covariates of detection and occupancy in that order.
data	unmarkedFrameOccu supplying data to the model.
K	the upper summation index used to numerically integrate out the latent abundance.
starts	initial values for the optimization.
method	Optimization method used by optim.
control	Other arguments passed to optim.
se	logical specifying whether or not to compute standard errors.

Details

See unmarked for detailed descriptions of passing data y, covdata.site, and covdata.obs, and specifying covariates with stateformula and detformula.

This function fits the latent abundance mixture model described in Royle and Nichols (2003).

The latent abundance of site i is modelled as Poisson:

$$N_i \sim Poisson(\lambda_i)$$

The detection of a single individual in site i during sample j is modelled as Bernoulli:

$$w_{ij} \sim Bernoulli(r_{ij})$$

.

Thus, the detection probability for a single site is linked to the detection probability for an individual by

$$p_{ij} = 1 - (1 - r_{ij})^{N_i}$$

Covariates of λ_i are modelled with the log link and covariates of r_{ij} are modelled with the logit link.

Value

unmarkedFit object describing the model fit.

Author(s)

Ian Fiske

References

Royle, J. A. and Nichols, J. D. (2003) Estimating Abundance from Repeated Presence-Absence Data or Point Counts. *Ecology*, 84(3) pp. 777–790.

Examples

```
data(birds)
woodthrushUMF <- unmarkedFrameOccu(woodthrush.bin)
# survey occasion-specific detection probabilities
(fm.wood.rn <- occuRN(~ obsNum ~ 1, woodthrushUMF))
```

`ovendata` *Removal data for the Ovenbird*

Description

Removal sampling data collected for the Ovenbird (*Seiurus aurocapillus*).

Usage

```
data(ovendata)
```

Format

The format is: chr "ovendata.list" which consists of

data matrix of removal counts

covariates data frame of site-level covariates

Source

J.A. Royle (see reference below)

References

Royle, J. A. (2004). Generalized estimators of avian abundance from count survey data. Animal Biodiversity and Conservation, 27(1), 375-386.

Examples

```
data(ovendata)
str(ovendata.list)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])), type = "removal")
```

parboot *Parametric bootstrap method for fitted models inheriting class.*

Description

Simulate datasets from a fitted model, refit the model, and generate a sampling distribution for a user-specified fit-statistic.

Arguments

object	a fitted model inheriting class "unmarkedFit"
statistic	a function returning a vector of fit-statistics. First argument must be the fitted model. Default is sum of squared residuals.
nsim	number of bootstrap replicates
report	print fit statistic every 'report' iterations during resampling
...	Additional arguments to be passed to statistic

Details

This function simulates datasets based upon a fitted model, refits the model, and evaluates a user-specified fit-statistic for each simulation. Comparing this sampling distribution to the observed statistic provides a means of evaluating goodness-of-fit or assessing uncertainty in a quantity of interest.

Value

An object of class parboot with three slots:

call	parboot call
t0	Numeric vector of statistics for original fitted model.
t.star	nsim by length(t0) matrix of statistics for each simulation fit.

Author(s)

Richard Chandler <rchandler@nrc.umass.edu>

Examples

```
data(linetran)
(dbreaksLine <- c(0, 5, 10, 15, 20))
lengths <- linetran$Length

ltUMF <- with(linetran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
siteCovs = data.frame(Length, area, habitat), dist.breaks = dbreaksLine,
tlength = lengths*1000, survey = "line", unitsIn = "m")
```

```
    })

# Fit a model
(fm <- distsamp(~area ~habitat, ltUMF))

# Function returning two fit-stats: sum of squared errors and population size at
# sampled plots.
fitStats <- function(fit) {
    sse <- SSE(fit)
    plot.area.ha <- lengths*1000 * 40 / 10000
    N <- sum(predict(fit, type="state")$Predicted*plot.area.ha, na.rm=TRUE)
    return(c(sse, N.hat=N))
    }

(pb <- parboot(fm, fitStats, nsim=25))
plot(pb, main="")
```

`pcount` *Fit the N-mixture point count model...*

Description

Fit the N-mixture point count model

Usage

```
pcount(formula, data, K, mixture=c("P", "NB"), starts, method="BFGS",
    control=list(), se=TRUE)
```

Arguments

`formula`	Double right-hand side formula describing covariates of detection and abundance, in that order
`data`	an unmarkedFramePCount object supplying data to the model.
`K`	Integer upper index of integration for N-mixture.
`mixture`	character specifying mixture: either "P" or "NB".
`starts`	vector of starting values
`method`	Optimization method used by `optim`.
`control`	Other arguments passed to `optim`.
`se`	logical specifying whether or not to compute standard errors.

Details

This function fits binomial-Poisson mixture model for spatially replicated point count data.

See `unmarkedFrame` for a description of how to supply by creating and unmarkedFrame.

This function fits the latent N-mixture model for point count data (Royle 2004, Kéry et al 2005).

The latent abundance distribution, $f(N|\theta)$ can be set as either a Poisson or a negative binomial random variable, depending on the setting of the `mixture` argument. `mixture = "P"` or `mixture = "NB"` select the Poisson or negative binomial distribution respectively. The mean of N_i is λ_i. If $N_i \sim NB$, then an additional parameter, α, describes dispersion (lower α implies higher variance).

The detection process is modeled as binomial: $y_{ij} \sim Binomial(N_i, p_{ij})$.

Covariates of λ_i use the log link and covariates of p_{ij} use the logit link.

Value

unmarkedFit object describing the model fit.

Author(s)

Ian Fiske <ianfiske@gmail.com>

References

Royle, J. A. (2004) N-Mixture Models for Estimating Population Size from Spatially Replicated Counts. *Biometrics* 60, pp. 108–105.

Kéry, M., Royle, J. A., and Schmid, H. (2005) Modeling Avaian Abundance from Replicated Counts Using Binomial Mixture Models. *Ecological Applications* 15(4), pp. 1450–1461.

Examples

```
data(mallard)
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,
obsCovs = mallard.obs)
(fm.mallard <- pcount(~ ivel+ date + I(date^2) ~ length + elev + forest, mallardUMF))
(fm.mallard.nb <- pcount(~ date + I(date^2) ~ length + elev, mixture = "NB", mallardUMF))
```

---

piFuns *Compute multinomial cell probabilities*

Description

Compute the cell probabilities used in the multinomial-Poisson model multinomPois.

Usage

```
removalPiFun(p)
doublePiFun(p)
```

Arguments

p	matrix of detection probabilities at each site for each observation

Details

These two functions are provided as examples of possible functions to calculate multinomial cell probabilities. Users may write their own functions for specific sampling designs and use them in multinomPois. See the example for a user-defined function.

Value

For removalPiFun, a matrix of cell probabilties for each site and sampling period.

For doublePiFun, a matrix of cell probabilities for each site and observer combination. Column one is probability observer 1 but not observer 2 detects the object, column two is probability that observer 2 but not observer 1 detects the object, and column 3 is probability of both detecting.

Examples

```
(pRem <- matrix(0.5, nrow=3, ncol=3)) # Capture probabilities
removalPiFun(pRem) # Cell probs

(pDouble <- matrix(0.5, 3, 2)) # Observer detection probs
doublePiFun(pDouble) # Cell probs

# A user-defined piFun calculating removal probs when time intervals differ.
# Here 10-minute counts were divided into 2, 3, and 5 minute intervals.
# This function could be supplied to unmarkedFrameMPois along with the obsToY
# argument shown below.

instRemPiFun <- function(p) {
M <- nrow(p)
J <- ncol(p)
pi <- matrix(NA, M, J)
p[,1] <- pi[,1] <- 1 - (1 - p[,1])^2
p[,2] <- 1 - (1 - p[,2])^3
p[,3] <- 1 - (1 - p[,3])^5
for(i in 2:J) {
pi[,i] <- pi[, i - 1]/p[, i - 1] * (1 - p[, i - 1]) * p[, i]
}
return(pi)
}

instRemPiFun(pRem)

# Associated obsToY matrix required by unmarkedFrameMPois
o2y <- diag(3)
o2y[upper.tri(o2y)] <- 1
o2y
```

pointtran *Simulated point-transect data*

Description

Response matrix of animals detected in five distance classes plus two covariates.

Usage

```
data(pointtran)
```

Format

A data frame with 30 observations on the following 7 variables.

`dc1` Counts in distance class 1 [0-5 m)

`dc2` Counts in distance class 2 [5-10 m)

`dc3` Counts in distance class 3 [10-15 m)

`dc4` Counts in distance class 4 [15-20 m)

`dc5` Counts in distance class 5 [20-25 m)

`area` a numeric vector

`habitat` a factor with levels `A B C`

Examples

```
data(pointtran)
pointtran

# Format for distsamp()
ptUMF <- with(pointtran, {
        unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4, dc5),
        siteCovs = data.frame(area, habitat),
        dist.breaks = seq(0, 25, by=5), survey = "point", unitsIn = "m")
        })
```

SE-methods *Methods for Function SE in Package 'unmarked'*

Description

Extract standard errors of parameter estimates from a fitted model.

Methods

obj = "linCombOrBackTrans" A model prediction

obj = "unmarkedEstimate" See `unmarkedEstimate-class`

obj = "unmarkedFit" A fitted model

sight2perpdist *Convert sight distance and sight angle to perpendicular distance.*

Description

When distance data are collected on line transects using sight distances and sight angles, they need to be converted to perpendicular distances before analysis.

Usage

```
sight2perpdist(sightdist, sightangle)
```

Arguments

sightdist	Distance from observer
sightangle	Angle from center line. In degrees between 0 and 180.

Value

Perpendicular distance

See Also

distsamp

Examples

```
round(sight2perpdist(10, c(0, 45, 90, 135, 180)))
```

simulate-methods *Methods for Function simulate in Package 'unmarked'*

Description

Simulate data from a fitted model.

Usage

```
## S4 method for signature 'unmarkedFitColExt':
simulate(object, nsim, seed, na.rm)
## S4 method for signature 'unmarkedFitDS':
simulate(object, nsim, seed, na.rm)
## S4 method for signature 'unmarkedFitMPois':
simulate(object, nsim, seed, na.rm)
## S4 method for signature 'unmarkedFitOccu':
simulate(object, nsim, seed, na.rm)
```

```
## S4 method for signature 'unmarkedFitOccuRN':
simulate(object, nsim, seed, na.rm)
## S4 method for signature 'unmarkedFitPCount':
simulate(object, nsim, seed, na.rm)
```

Arguments

object	Fitted model of appropriate S4 class
nsim	Number of simulations
seed	Seed for random number generator. Not currently implemented
na.rm	Logical, should missing values be removed?

Methods

object = "unmarkedFitColExt" A model fit by colext

object = "unmarkedFitDS" A model fit by distsamp

object = "unmarkedFitMPois" A model fit by multinomPois

object = "unmarkedFitOccu" A model fit by occu

object = "unmarkedFitOccuRN" A model fit by occuRN

object = "unmarkedFitPCount" A model fit by pcount

SSE *Compute Sum of Squared Residuals for a Model Fit.*

Description

Compute the sum of squared residuals for an unmarked fit object. This is useful for a parboot.

Usage

```
SSE(fit)
```

Arguments

fit	An unmarked fit object.

Value

A numeric value for the models SSE.

See Also

parboot

`unmarkedEstimate-class` *Class "unmarkedEstimate"*

Description

Contains parameter estimates, covariance matrix, and metadata

Objects from the Class

Creating these objects is done internally not by users.

Slots

`name`: Object of class `"character"` storing parameter names

`short.name`: Object of class `"character"` storing abbreviated parameter names

`estimates`: Object of class `"numeric"`

`covMat`: Object of class `"matrix"`

`covMatBS`: Object of class `"matrix"`

`invlink`: Object of class `"character"`

`invlinkGrad`: Object of class `"character"`

Methods

backTransform `signature(obj = "unmarkedEstimate")`

coef `signature(object = "unmarkedEstimate")`

confint `signature(object = "unmarkedEstimate")`

linearComb `signature(obj = "unmarkedEstimate", coefficients = "matrixOrVector")`

SE `signature(obj = "unmarkedEstimate")`

show `signature(object = "unmarkedEstimate")`

vcov `signature(object = "unmarkedEstimate")`

Note

These methods are typically called within a call to a method for `unmarkedFit-class`

Examples

```
showClass("unmarkedEstimate")
```

unmarkedEstimateList-class *Class "unmarkedEstimateList"*

Description

Class to hold multiple unmarkedEstimates in an `unmarkedFit`

Slots

`estimates`: A `"list"` of models.

unmarkedFit-class *Class "unmarkedFit"*

Description

Contains fitted model information which can be manipulated or extracted using the methods described below.

Slots

`fitType`: Object of class `"character"`

`call`: Object of class `"call"`

`formula`: Object of class `"formula"`

`data`: Object of class `"unmarkedFrame"`

`sitesRemoved`: Object of class `"numeric"`

`estimates`: Object of class `"unmarkedEstimateList"`

`AIC`: Object of class `"numeric"`

`opt`: Object of class `"list"` containing results from `optim`

`negLogLike`: Object of class `"numeric"`

`nllFun`: Object of class `"function"`

`knownOcc`: unmarkedFitOccu only: sites known to be occupied

`K`: unmarkedFitPCount only: upper bound used in integration

`mixture`: unmarkedFitPCount only: Mixing distribution

`keyfun`: unmarkedFitDS only: detection function used by distsamp

`unitsOut`: unmarkedFitDS only: density units

Methods

[`signature(x = "unmarkedFit", i = "ANY", j = "ANY", drop = "ANY")`: extract one of names(obj), eg 'state' or 'det'

backTransform `signature(obj = "unmarkedFit")`: back-transform parameters to original scale when no covariate effects are modeled

coef `signature(object = "unmarkedFit")`: returns parameter estimates. type can be one of names(obj), eg 'state' or 'det'. If altNames=TRUE estimate names are more specific.

confint `signature(object = "unmarkedFit")`: Returns confidence intervals. Must specify type and method (either "normal" or "profile")

fitted `signature(object = "unmarkedFit")`: returns expected values of Y

getData `signature(object = "unmarkedFit")`: extracts data

getP `signature(object = "unmarkedFit")`: calculates and extracts expected detection probabilities

hessian `signature(object = "unmarkedFit")`: Returns hessian matrix

linearComb `signature(obj = "unmarkedFit", coefficients = "matrixOrVector")`: Returns estimate and SE on original scale when covariates are present

mle `signature(object = "unmarkedFit")`: Same as coef(fit)?

names `signature(x = "unmarkedFit")`: Names of parameter levels

nllFun `signature(object = "unmarkedFit")`: returns negative log-likelihood used to estimate parameters

parboot `signature(object = "unmarkedFit")`: Parametric bootstrapping method to assess goodness-of-fit

plot `signature(x = "unmarkedFit", y = "missing")`: Plots expected vs. observed values

predict `signature(object = "unmarkedFit")`: Returns predictions and standard errors for original data or for covariates in a new data.frame

profile `signature(fitted = "unmarkedFit")`: used by confint method='profile'

residuals `signature(object = "unmarkedFit")`: returns residuals

sampleSize `signature(object = "unmarkedFit")`: returns number of sites in sample

SE `signature(obj = "unmarkedFit")`: returns standard errors

show `signature(object = "unmarkedFit")`: concise results

summary `signature(object = "unmarkedFit")`: results with more details

update `signature(object = "unmarkedFit")`: refit model with changes to one or more arguments

vcov `signature(object = "unmarkedFit")`: returns variance-covariance matrix

smoothed `signature(object="unmarkedFitColExt")`: Returns the smoothed trajectory from a colonization-extinction model fit. Takes additional logical argument mean which specifies whether or not to return the average over sites.

projected `signature(object="unmarkedFitColExt")`: Returns the projected trajectory from a colonization-extinction model fit. Takes additional logical argument mean which specifies whether or not to return the average over sites.

logLik `signature(object="unmarkedFit")`: Returns the log-likelihood.

LRT `signature(m1="unmarkedFit", m2="unmarkedFit")`: Returns the chi-squared statistic, degrees-of-freedom, and p-value from a Likelihood Ratio Test.

Note

This is a superclass with child classes for each fit type

Examples

```
showClass("unmarkedFit")

# Format removal data for multinomPois
data(ovendata)
ovenFrame <- unmarkedFrameMPois(y = ovendata.list$data,
siteCovs = as.data.frame(scale(ovendata.list$covariates[,-1])),
type = "removal")

# Fit a model
(fm1 <- multinomPois(~ 1 ~ ufp + trba, ovenFrame))

# Apply a bunch of methods to the fitted model

names(fm1)
fm1['state']
fm1['det']

backTransform(fm1, type ='det')

coef(fm1, type='state')

confint(fm1, type='state', method='profile')

fitted(fm1)

getData(fm1)

getP(fm1)

# Return predicted abundance at specified covariate values
linearComb(fm1, c(Int = 1, ufp = 0, trba = 0), type='state')

# Assess goodness-of-fit
parboot(fm1)

plot(fm1)

# Predict abundance at specified covariate values.
newdat <- data.frame(ufp = 0, trba = seq(-1, 1, length=10))

predict(fm1, type='state', newdata=newdat)
```

```
sampleSize(fm1)

summary(fm1)

(fmNull <- update(fm1, formula = ~1 ~1))

vcov(fm1, type='state')
```

`unmarkedFitList-class` *Class "unmarkedFitList"*

Description

Class to hold multiple nested fitted models from one of `unmarked`'s fitting functions

Objects from the Class

Objects can be created by using the `fitList` function.

Slots

`fits`: A `"list"` of models.

Methods

modSel `signature(object = "unmarkedFitList")`: Model selection

predict `signature(object = "unmarkedFitList")`: Model-averaged prediction

Note

Model-averaging regression coefficients is intentionally not implemented.

See Also

`fitList`, `unmarkedFit`

Examples

```
showClass("unmarkedFitList")
```

`unmarkedFrame` *Create an unmarkedFrame, or one of its child classes.*

Description

Constructor for unmarkedFrames.

Usage

```
unmarkedFrame(y, siteCovs=NULL, obsCovs=NULL, mapInfo, obsToY)
```

Arguments

`y`	A matrix of the observed measured data.
`siteCovs`	`data.frame` of covariates that vary at the site level.
`obsCovs`	list of `data.frame`s of covariates that vary within sites.
`obsToY`	optional matrix specifying relationship between observation-level covariates and response matrix
`mapInfo`	geographic coordinate information

Details

unmarkedFrame is the S4 class that holds data structures to be passed to the model-fitting functions in unmarked.

An unmarkedFrame contains the observations (`y`), covariates measured at the observation level (`obsCovs`), and covariates measured at the site level (`siteCovs`). For a data set with M sites and J observations at each site, y is an M x J matrix. `obsCovs` and `siteCovs` are both data frames (see data.frame). `siteCovs` has M rows so that each row contains the covariates for the corresponding sites. `obsCovs` has M*obsNum rows so that each covariates is ordered by site first, then observation number. Missing values are coded with `NA` in any of y, siteCovs, or obsCovs.

Additionally, unmarkedFrames contain metadata: obsToY, mapInfo. ObsToY is a matrix describing relationship between response matrix and observation-level covariates. Generally this does not need to be supplied by the user; however, it may be needed when using `multinomPois`. For example, double observer sampling, y has 3 columns corresponding the observer 1, observer 2, and both, but there were only two independent observations. In this situation, y has 3 columns, but obsToY must be specified.

Several child classes of `unmarkedFrame` require addional metadata. For example, `unmarkedFrameDS` is used to organize distsance sampling data for the `distsamp` function, and it has arguments dist.breaks, tlength, survey, and unitsIn, which specify the distance interval cut points, transect lengths, "line" or "point" transect, and units of measure, respectively.

All site-level covariates are automatically copied to obsCovs so that site level covariates are available at the observation level.

Value

an unmarkedFrame object

See Also

unmarkedFrame-class

Examples

```
# Set up data for pcount()
data(mallard)
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,
obsCovs = mallard.obs)
summary(mallardUMF)


# Set up data for occu()
data(frogs)
pferUMF <- unmarkedFrameOccu(pfer.bin)


# Set up data for distsamp()
data(linetran)
ltUMF <- with(linetran, {
unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4),
siteCovs = data.frame(Length, area, habitat),
dist.breaks = c(0, 5, 10, 15, 20),
tlength = linetran$Length * 1000, survey = "line", unitsIn = "m")
})
summary(ltUMF)


# Set up data for multinomPois()
data(ovendata)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
siteCovs=as.data.frame(scale(ovendata.list$covariates[,-1])),
type = "removal")
summary(ovenFrame)


# Set up data for colext()
frogUMF <- formatMult(masspcru)
summary(frogUMF)
```

---

unmarkedFrame-class

Class "unmarkedFrame"

Description

Methods for manipulating, summarizing and viewing unmarkedFrames

Objects from the Class

Objects can be created by calls to the constructor function `unmarkedFrame`. These objects are passed to the data argument of the fitting functions.

Slots

`y`: Object of class `"matrix"`

`obsCovs`: Object of class `"optionalDataFrame"`

`siteCovs`: Object of class `"optionalDataFrame"`

`mapInfo`: Object of class `"optionalMapInfo"`

`obsToY`: Object of class `"optionalMatrix"`

Methods

[`signature(x = "unmarkedFrame", i = "numeric", j = "missing", drop = "missing")`: ...

[`signature(x = "unmarkedFrame", i = "numeric", j = "numeric", drop = "missing")`: ...

[`signature(x = "unmarkedFrame", i = "missing", j = "numeric", drop = "missing")`: ...

coordinates `signature(object = "unmarkedFrame")`: extract coordinates

getY `signature(object = "unmarkedFrame")`: extract y matrix

numSites `signature(object = "unmarkedFrame")`: extract M

numY `signature(object = "unmarkedFrame")`: extract ncol(y)

obsCovs `signature(object = "unmarkedFrame")`: extract observation-level covariates

obsCovs<- `signature(object = "unmarkedFrame")`: add or modify observation-level covariates

obsNum `signature(object = "unmarkedFrame")`: extract number of observations

obsToY `signature(object = "unmarkedFrame")`:

obsToY<- `signature(object = "unmarkedFrame")`: ...

plot `signature(x = "unmarkedFrame", y = "missing")`: visualize response variable.
Takes additional argument `panels` which specifies how many panels data should be split over.

projection `signature(object = "unmarkedFrame")`: extract projection information

show `signature(object = "unmarkedFrame")`: view data as data.frame

siteCovs `signature(object = "unmarkedFrame")`: extract site-level covariates

siteCovs<- `signature(object = "unmarkedFrame")`: add or modify site-level covariates

summary `signature(object = "unmarkedFrame")`: summarize data

Note

This is a superclass with child classes for each fitting function

See Also

unmarkedFrame, unmarkedFit, unmarked-package

Examples

```
# Organize data for pcount()
data(mallard)
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,
obsCovs = mallard.obs)


# Vizualize it
plot(mallardUMF)

mallardUMF


# Summarize it
summary(mallardUMF)

str(mallardUMF)

numSites(mallardUMF)

numY(mallardUMF)

obsNum(mallardUMF)


# Extract components of data
getY(mallardUMF)

obsCovs(mallardUMF)
obsCovs(mallardUMF, matrices = TRUE)

siteCovs(mallardUMF)

mallardUMF[1:5,] # First 5 rows in wide format

mallardUMF[,1:2] # First 2 observations
```

`unmarkedMultFrame` *Create an unmarkedMultFrame or an unmarkedFrameGMM.*

Description

These functions construct flavors of unmarkedFrames for data collected during primary and secondary sampling periods.

Usage

```
unmarkedMultFrame(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs)
unmarkedFrameGMM(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs,
    type, obsToY, piFun)
```

Arguments

`y`	A matrix of the observed data.
`siteCovs`	Data frame of covariates that vary at the site level.
`obsCovs`	Data frame of covariates that vary within site-year-observation level.
`numPrimary`	Number of primary time periods (seasons in the multiseason model).
`yearlySiteCovs`	Data frame containing covariates at the site-year level.
`type`	Either "removal" or "double" for constant-interval removal sampling or double observer sampling. This should be not be specified for other types of survey designs.
`obsToY`	A matrix specifying relationship between observation-level covariates and response matrix
`piFun`	A function converting an MxJ matrix of detection probabilities into an MxJ matrix of multinomial cell probabilities.

Details

unmarkedMultFrame objects are used by `colext`.

unmarkedFrameGMM objects are used by `gmultmix`.

For a study with M sites, T years, and a maximum of J observations per site-year, the data are shaped as follows. `y` is an $M \times TJ$ matrix, with each row corresponding to a site. `siteCovs` is a data frame with M rows. `yearlySiteCovs` is a data frame with MT rows which are in site-major, year-minor order. `obsCovs` is a data frame with MTJ rows, which are ordered by site-year-observation, so that a column of `obsCovs` corresponds to `as.vector(t(y))`, element-by-element. The number of years must be specified in `numPrimary`.

If the data are in long format, the convenience function `formatMult` is useful for creating the unmarkedMultFrame.

unmarkedFrameGMM is a superclass of unmarkedMultFrame containing information on the the survey design used that resulted in multinomial outcomes. For constant-interval removal sampling,

you can set type="removal" and ignore the arguments obsToY and piFun. Similarly, for double-observer sampling, setting type="double" will automatically create an appropiate obsToY matrix and piFuns. For all other situations, the type argument should be ignored and the obsToY and piFun arguments must be specified. piFun must be a function that converts an MxJ matrix of detection probabilities into an MxJ matrix of multinomial cell probabilities. obsToY is a matrix describing how the obsCovs relate to the observed counts y. For further discussion and examples see the help page for multinomPois and piFuns.

An unmarkedFrameGMM object can be created from an unmarkedMultFrame using the "as" conversion method. See examples.

Value

an unmarkedMultFrame or unmarkedFrameGMM object

Examples

```
n <- 50   # number of sites
T <- 4    # number of primary periods
J <- 3    # number of secondary periods

site <- 1:50
years <- data.frame(matrix(rep(2010:2013, each=n), n, T))
years <- data.frame(lapply(years, as.factor))
occasions <- data.frame(matrix(rep(1:(J*T), each=n), n, J*T))

y <- matrix(0:1, n, J*T)

umf <- unmarkedMultFrame(y=y,
    siteCovs = data.frame(site=site),
    obsCovs=list(occasion=occasions),
    yearlySiteCovs=data.frame(year=years),
    numPrimary=T)

umfGMM1 <- unmarkedFrameGMM(y=y,
    siteCovs = data.frame(site=site),
    obsCovs=list(occasion=occasions),
    yearlySiteCovs=data.frame(year=years),
    numPrimary=T, type="removal")


# A user-defined piFun calculating removal probs when time intervals differ.
instRemPiFun <- function(p) {
M <- nrow(p)
J <- ncol(p)
pi <- matrix(NA, M, J)
p[,1] <- pi[,1] <- 1 - (1 - p[,1])^2
p[,2] <- 1 - (1 - p[,2])^3
p[,3] <- 1 - (1 - p[,3])^5
for(i in 2:J) {
pi[,i] <- pi[, i - 1]/p[, i - 1] * (1 - p[, i - 1]) * p[, i]
```

```
}
return(pi)
}

# Associated obsToY matrix required by unmarkedFrameMPois
o2y <- diag(ncol(y))
o2y[upper.tri(o2y)] <- 1
o2y


umfGMM2 <- unmarkedFrameGMM(y=y,
    siteCovs = data.frame(site=site),
    obsCovs=list(occasion=occasions),
    yearlySiteCovs=data.frame(year=years),
    numPrimary=T, obsToY=o2y, piFun="instRemPiFun")

str(umfGMM2)
```

---

vcov-methods *Methods for Function vcov in Package 'unmarked'*

Description

Extract variance-covariance matrix from a fitted model.

Methods

object = "linCombOrBackTrans" See linearComb-methods

object = "unmarkedEstimate" See unmarkedEstimate-class

object = "unmarkedFit" A fitted model

[-methods *Methods for bracket extraction [in Package 'unmarked'*

Description

Methods for bracket extraction [in Package 'unmarked'

Usage

```
## S4 method for signature 'unmarkedEstimateList,ANY,ANY,ANY':
[(x, i, j, drop)
## S4 method for signature 'unmarkedFit,ANY,ANY,ANY':
[(x, i, j, drop)
## S4 method for signature 'unmarkedFrame,numeric,numeric,missing':
[(x, i, j)
## S4 method for signature 'unmarkedMultFrame,missing,numeric,missing':
[(x, i, j)
```

Arguments

x	Object of appropriate S4 class
i	Row numbers
j	Observation numbers (eg occasions, distance classes, etc...)
drop	Not currently used

Methods

x = "unmarkedEstimateList", i = "ANY", j = "ANY", drop = "ANY" Extract a unmarkedEstimate object from an unmarkedEstimateList by name (either 'det' or 'state')

x = "unmarkedFit", i = "ANY", j = "ANY", drop = "ANY" Extract a unmarkedEstimate object from an unmarkedFit by name (either 'det' or 'state')

x = "unmarkedFrame", i = "missing", j = "numeric", drop = "missing" Extract observations from an unmarkedFrame.

x = "unmarkedFrame", i = "numeric", j = "missing", drop = "missing" Extract rows from an unmarkedFrame

x = "unmarkedFrame", i = "numeric", j = "numeric", drop = "missing" Extract rows and observations from an unmarkedFrame

x = "unmarkedMultFrame", i = "missing", j = "numeric", drop = "missing" Extract primary sampling periods from an unmarkedMultFrame

x = "unmarkedFrame", i = "list", j = "missing", drop = "missing" List is the index of observations to subset for each site.

Examples

```
data(mallard)
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,
obsCovs = mallard.obs)
summary(mallardUMF)

mallardUMF[1:5,]
mallardUMF[,1:2]
mallardUMF[1:5, 1:2]
```

Index