

Package ‘unmarked’

March 29, 2012

Version 0.9-7

Date 2012-03-29

Type Package

Title Models for Data from Unmarked Animals

Author Ian Fiske, Richard Chandler, Andy Royle, Marc Kery

Maintainer Richard Chandler <rchandler@usgs.gov>

Depends R (>= 2.12.0), methods, reshape, lattice, Rcpp (>= 0.8.0),RcppArmadillo (>= 0.2.0)

Suggests raster

Imports graphics, stats, utils

Description Unmarked fits hierarchical models of animal abundance and occurrence to data collected using survey methods such as point counts, site occupancy sampling, distance sampling, removal sampling, and double observer sampling. Parameters governing the state and observation processes can be modeled as functions of covariates.

License GPL (>= 3)

LazyLoad yes

LazyData yes

Collate

‘classes.R’ ‘unmarkedEstimate.R’ ‘mapInfo.R’ ‘unmarkedFrame.R’ ‘unmarkedFit.R’ ‘utils.R’ ‘getDesign.R’ ‘colect.R’ ‘dis

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

SystemRequirements GNU make

URL <http://groups.google.com/group/unmarked>,
<https://sites.google.com/site/unmarkedinfo/home>,<http://github.com/ianfiske/unmarked>,
<http://github.com/rbchan/unmarked>

Repository CRAN

Date/Publication 2012-03-29 21:00:00

R topics documented:

unmarked-package	3
backTransform-methods	7
birds	8
coef-methods	8
colext	9
confint-methods	12
csvToUMF	13
detFuns	14
distsamp	15
fitList	18
fitted-methods	19
formatDistData	20
formatMult	22
formatWideLong	23
frogs	24
gdistsamp	25
getP-methods	27
gf	28
gmultmix	29
imputeMissing	31
lambda2psi	32
linearComb-methods	33
linetran	33
mallard	34
masspcru	35
modSel	36
multinomPois	37
nonparboot-methods	39
occu	40
occuRN	42
ovendata	44
parboot	45
pcount	46
pcountOpen	49
piFuns	52
pointtran	53
predict-methods	54
ranef-methods	55
SE-methods	57
sight2perpdist	57
simulate-methods	58
SSE	59
unmarkedEstimate-class	59
unmarkedEstimateList-class	60
unmarkedFit-class	60
unmarkedFitList-class	63

unmarkedFrame 65
 unmarkedFrame-class 67
 unmarkedFrameDS 69
 unmarkedFrameMPois 70
 unmarkedFrameOccu 72
 unmarkedFramePCO 74
 unmarkedFramePCount 76
 unmarkedMultFrame 78
 unmarkedRanef-class 81
 vcov-methods 82
 [-methods 82

Index 84

unmarked-package *Models for Data from Unmarked Animals*

Description

unmarked fits hierarchical models of animal occurrence and abundance to data collected on species subject to imperfect detection. Examples include single- and multi-season site occupancy models, binomial N-mixture models, and multinomial N-mixture models. The data can arise from survey methods such as occurrence sampling, temporally replicated counts, removal sampling, double observer sampling, and distance sampling. Parameters governing the state and observation processes can be modeled as functions of covariates. General treatment of these models can be found in MacKenzie et al. (2006) and Royle and Dorazio (2008). The primary reference for the package is Fiske and Chandler (2011).

Details

Package: unmarked
 Type: Package
 Version: 0.9-7
 License: GPL (>= 3)

Overview of Model-fitting Functions:

[occu](#) fits occurrence models with no linkage between abundance and detection (MacKenzie et al. 2002).

[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 (aka binomial mixture models) to repeated count data (Royle 2004a, Kéry et al 2005).

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

`gdistsamp` fits the generalized distance sampling model described by Chandler et al. (2011) to distance data recorded in discrete intervals.

`multinomPois` fits the multinomial-Poisson model of Royle (2004b) to data collected using methods such as removal sampling or double observer sampling.

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

`pcountOpen` fits the open population model of Dail and Madsen (2011) to repeated count data. This is a generalized form of the Royle (2004a) N-mixture model that includes parameters for recruitment and apparent survival.

Data: All data are 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: `unmarked`'s model-fitting functions allow specification of covariates for both the state process and the detection process. For two-level hierarchical models, (eg `occu`, `occuRN`, `pcount`, `multinomPois`, `distsamp`) 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

$$\tilde{\tilde{x}}_1 + x_2 + \dots + x_n \tilde{x}_1 + x_2 + \dots + x_n$$

where x_1 through x_n are additive covariates of the process of interest. Using two tildes in a single formula differs from standard R convention, but it is informative about the model being fit. 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. For models with more than two processes (eg `colext`, `gmultmix`, `pcountOpen`), single right-hand sided formulas (only one tilde) are used to model each parameter.

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

Author(s)

Ian Fiske, Richard Chandler, Andy Royle, and Marc Kéry

References

- Dail, D. and L. Madsen. 2011. Models for estimating abundance from repeated counts of an open metapopulation. *Biometrics* 67:577-587.
- Fiske, I. and R. B. Chandler. 2011. `unmarked`: An R package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software* 43:1-23.
- Kéry, M., Royle, J. A., and Schmid, H. 2005 Modeling avian abundance from replicated counts using binomial mixture models. *Ecological Applications* 15:1450-1461.
- MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. A. Royle, and C. A. Langtimm. 2002. Estimating site occupancy rates when detection probabilities are less than one. *Ecology* 83: 2248-2255.

- MacKenzie, D. I., J. D. Nichols, J. E. Hines, M. G. Knutson, and A. B. Franklin. 2003. Estimating site occupancy, colonization, and local extinction when a species is detected imperfectly. *Ecology* 84:2200–2207.
- MacKenzie, D. I., J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines. 2006. *Occupancy Estimation and Modeling*. Amsterdam: Academic Press.
- Royle, J. A. 2004a. N-Mixture models for estimating population size from spatially replicated counts. *Biometrics* 60:108–105.
- Royle, J. A. 2004b. Generalized estimators of avian abundance from count survey data. *Animal Biodiversity and Conservation* 27:375–386.
- Royle, J. A., D. K. Dawson, and S. Bates. 2004. Modeling abundance effects in distance sampling. *Ecology* 85:1591–1597.
- Royle, J. A., and R. M. Dorazio. 2006. Hierarchical models of animal abundance and occurrence. *Journal Of Agricultural Biological And Environmental Statistics* 11:249–263.
- Royle, J. A. and R. M. Dorazio. 2008. *Hierarchical Modeling and Inference in Ecology*. Academic Press.
- Royle, J. A. and J. D. Nichols. 2003. Estimating Abundance from Repeated Presence-Absence Data or Point Counts. *Ecology*, 84:777–790.

Examples

```
## An example site-occupancy analysis

# Simulate occupancy data
set.seed(344)
nSites <- 100
nReps <- 5
covariates <- data.frame(veght=rnorm(nSites),
  habitat=factor(c(rep('A', 50), rep('B', 50))))

psipars <- c(-1, 1, -1)
ppars <- c(1, -1, 0)
X <- model.matrix(~veght+habitat, covariates) # design matrix
psi <- plogis(X %*% psipars)
p <- plogis(X %*% ppars)

y <- matrix(NA, nSites, nReps)
z <- rbinom(nSites, 1, psi) # true occupancy state
for(i in 1:nSites) {
  y[i,] <- rbinom(nReps, 1, z[i]*p[i])
}

# Organize data and look at it
umf <- unmarkedFrameOccu(y = y, siteCovs = covariates)
head(umf)
summary(umf)

# Fit some models
```

```

fm1 <- occu(~1 ~1, umf)
fm2 <- occu(~veght+habitat ~veght+habitat, umf)
fm3 <- occu(~veght ~veght+habitat, umf)

# Model selection
fms <- fitList(m1=fm1, m2=fm2, m3=fm3)
modSel(fms)

# Empirical Bayes estimates of the number of sites occupied
sum(bup(ranef(fm3), stat="mode"))      # Sum of posterior modes
colSums(confinf(ranef(fm3)))          # 95% CI
sum(z)                                # Actual

# Model-averaged prediction and plots

# psi in each habitat type
newdata1 <- data.frame(habitat=c('A', 'B'), veght=0)
Epsi1 <- predict(fms, type="state", newdata=newdata1)
with(Epsi1, {
  plot(1:2, Predicted, xaxt="n", xlim=c(0.5, 2.5), ylim=c(0, 0.5),
       xlab="Habitat",
       ylab=expression(paste("Probability of occurrence (", psi, ")")),
       cex.lab=1.2,
       pch=16, cex=1.5)
  axis(1, 1:2, c('A', 'B'))
  arrows(1:2, Predicted-SE, 1:2, Predicted+SE, angle=90, code=3, length=0.05)
})

# psi and p as functions of vegetation height
newdata2 <- data.frame(habitat=factor('A', levels=c('A', 'B')),
                      veght=seq(-2, 2, length=50))
Epsi2 <- predict(fms, type="state", newdata=newdata2, appendData=TRUE)
Ep <- predict(fms, type="det", newdata=newdata2, appendData=TRUE)

op <- par(mfrow=c(2, 1), mai=c(0.9, 0.8, 0.2, 0.2))
plot(Predicted~veght, Epsi2, type="l", lwd=2, ylim=c(0,1),
     xlab="Vegetation height (standardized)",
     ylab=expression(paste("Probability of occurrence (", psi, ")")))
lines(lower ~ veght, Epsi2, col=gray(0.7))
lines(upper ~ veght, Epsi2, col=gray(0.7))
plot(Predicted~veght, Ep, type="l", lwd=2, ylim=c(0,1),
     xlab="Vegetation height (standardized)",
     ylab=expression(paste("Detection probability (", italic(p), ")")))
lines(lower~veght, Ep, col=gray(0.7))
lines(upper~veght, Ep, col=gray(0.7))
par(op)

```

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.

Examples

```
data(mallard)  
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,  
  obsCovs = mallard.obs)  
  
(fm <- pcount(~ 1 ~ forest, mallardUMF)) # Fit a model  
backTransform(fm, type="det") # This works because there are no detection covariates  
#backTransform(fm, type="state") # This doesn't work because covariates are present  
lc <- linearComb(fm, c(1, 0), type="state") # Estimate abundance on the log scale when forest=0  
backTransform(lc) # Abundance on the original scale
```

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", 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 .
se	logical specifying whether or not to compute standard errors.
...	Additional arguments to optim, such as lower and upper bounds

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., J. D. Nichols, J. E. Hines, M. G. Knutson, and A. B. Franklin. 2003. Estimating site occupancy, colonization, and local extinction when a species is detected imperfectly. *Ecology* 84:2200–2207.

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

```
# Fake data
R <- 4 # number of sites
J <- 3 # number of secondary sampling occasions
T <- 2 # number of primary periods

y <- matrix(c(
  1,1,0, 0,0,0,
  0,0,0, 0,0,0,
  1,1,1, 1,1,0,
  1,0,1, 0,0,1), nrow=R, ncol=J*T, byrow=TRUE)
y

site.covs <- data.frame(x1=1:4, x2=factor(c('A', 'B', 'A', 'B')))
```

```

site.covs

yearly.site.covs <- list(
  year = matrix(c(
    'year1', 'year2',
    'year1', 'year2',
    'year1', 'year2',
    'year1', 'year2'), nrow=R, ncol=T, byrow=TRUE)
  )
yearly.site.covs

obs.covs <- list(
  x4 = matrix(c(
    -1,0,1, -1,1,1,
    -2,0,0, 0,0,2,
    -3,1,0, 1,1,2,
    0,0,0, 0,1,-1), nrow=R, ncol=J*T, byrow=TRUE),
  x5 = matrix(c(
    'a','b','c', 'a','b','c',
    'd','b','a', 'd','b','a',
    'a','a','c', 'd','b','a',
    'a','b','a', 'd','b','a'), nrow=R, ncol=J*T, byrow=TRUE))
obs.covs

umf <- unmarkedMultFrame(y=y, siteCovs=site.covs,
  yearlySiteCovs=yearly.site.covs, obsCovs=obs.covs,
  numPrimary=2) # organize data
umf # look at data
summary(umf) # summarize
fm <- colext(~1, ~1, ~1, ~1, umf) # fit a model
fm

# Real data
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)

# Empirical Bayes estimates of number of sites occupied in each year

```

```

re <- ranef(fm)
modes <- colSums(bup(re, stat="mode"))
CI <- colSums(confint(re))
plot(1:7, modes, xlab="Year", ylab="Sites occupied", ylim=c(0, 70))
arrows(1:7, CI[1,], 1:7, CI[2,], code=3, angle=90, length=0.05)

## 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 asymptotic 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 appropriately formatted 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 <i>Details</i>)
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)
drex(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", 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 <code>optim</code> .
se	logical specifying whether or not to compute standard errors.
...	Additional arguments to <code>optim</code> , such as lower and upper bounds

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 π_{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 <rchandler@usgs.gov>

References

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

See Also

[unmarkedFrameDS](#), [unmarkedFit-class fitList](#), [formatDistData](#), [parboot](#), [sight2perpdist](#), [detFuns](#), [gdistsamp](#), [ranef](#). 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 det covars
# Empirical Bayes estimates of posterior distribution for N_i
plot(ranef(fm1, K=50))

# Effective strip half-width
(eshw <- integrate(gxhn, 0, 20, sigma=10.9)$value)

# Detection probability
eshw / 20 # 20 is strip-width

# 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)")

# effective radius
sig <- exp(coef(fmp1, type="det"))
ea <- 2*pi * integrate(grhn, 0, 25, sigma=sig)$value # effective area
sqrt(ea / pi) # effective radius

# detection probability
ea / (pi*25^2)

## End(Not run)

```

fitList

constructor of unmarkedFitList objects

Description

Organize models for model selection or model-averaged prediction.

Usage

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

Arguments

...	Fitted models. Preferably named.
fits	An alternative way of providing the models. A (preferably 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@usgs.gov>

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)

## 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)

# Extract coefficients and standard errors
coef(fmList)
SE(fmList)

# 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	<i>Bin distance data</i>
----------------	--------------------------

Description

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

Usage

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

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, name of the column in distData that contains the distances. The distances should be numeric.
transectNameCol	character, column name containing transect names. The transect column should be a factor.
dist.breaks	numeric vector of distance interval cutpoints. Length must equal J+1.
occasionCol	optional character. If transects were visited more than once, this can be used to format data for <code>gdistsamp</code> . It is the name of the column in distData that contains the occasion numbers. The occasion column should be a factor.

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. Alternatively, if each transect was surveyed T times, the resulting matrix is M x JT, which is the format required by `gdistsamp`, see `unmarkedFrameGDS`.

Value

An M x J or M x JT matrix containing the binned distance data. Transect names will become rownames and colnames will describe the distance intervals.

Note

It is important that the factor containing transect names includes levels for all the transects surveyed, not just those with ≥ 1 detection. Likewise, if transects were visited more than once, the factor containing the occasion numbers should include levels for all occasions. 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(0, 8, 10, 12, 14, 18)

# Create formatted response matrix
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

## Example for data from multiple occasions

dat2 <- data.frame(distance=1:100, site=gl(5, 20),
                  visit=factor(rep(1:4, each=5)))
cutpt <- seq(0, 100, by=25)
y2 <- formatDistData(dat2, "distance", "site", cutpt, "visit")
umf <- unmarkedFrameGDS(y=y2, numPrimary=4, survey="point",
                      dist.breaks=cutpt, unitsIn="m")

```

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	<i>Convert between wide and long data formats.</i>
----------------	--

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 separator 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	<i>2001 Delaware North American Amphibian Monitoring Program Data</i>
-------	---

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)
```

gdistsamp

Fit the generalized distance sampling model of Chandler et al. (2011).

Description

Extends the distance sampling model of Royle et al. (2004) to estimate the probability of being available for detection. Also allows abundance to be modeled using the negative binomial distribution.

Usage

```
gdistsamp(lambdaformula, phiformula, pformula, data, keyfun =
c("halfnorm", "exp", "hazard", "uniform"), output = c("abund",
"density"), unitsOut = c("ha", "kmsq"), mixture = c("P", "NB"), K,
starts, method = "BFGS", se = TRUE, rel.tol=1e-4, ...)
```

Arguments

lambdaformula	A right-hand side formula describing the abundance covariates.
phiformula	A right-hand side formula describing the availability covariates.
pformula	A right-hand side formula describing the detection function covariates.
data	An object of class unmarkedFrameGDS
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.
mixture	Either "P" or "NB" for the Poisson and negative binomial models of abundance.
K	An integer value specifying the upper bound used in the integration.
starts	A numeric vector of starting values for the model parameters.
method	Optimization method used by optim .
se	logical specifying whether or not to compute standard errors.
rel.tol	relative accuracy for the integration of the detection function. See integrate . You might try adjusting this if you get an error message related to the integral. Alternatively, try providing different starting values.
...	Additional arguments to optim , such as lower and upper bounds

Details

This model extends the model of Royle et al. (2004) by estimating the probability of being available for detection ϕ . This effectively relaxes the assumption that $g(0) = 1$. In other words, individuals at a distance of 0 are not assumed to be detected with certainty. To estimate this additional parameter, replicate distance sampling data must be collected at each transect. Thus the data are collected at $i = 1, 2, \dots, R$ transects on $t = 1, 2, \dots, T$ occasions. As with the model of Royle et al. (2004), the detections must be binned into distance classes. These data must be formatted in a matrix with R rows, and JT columns where J is the number of distance classes. See [unmarkedFrameGDS](#) for more information.

Value

An object of class `unmarkedFitGDS`.

Note

If you aren't interested in estimating ϕ , but you want to use the negative binomial distribution, simply set `numPrimary=1` when formatting the data.

Note

You cannot use `obsCovs`, but you can use `yearlySiteCovs` (a confusing name since this model isn't for multi-year data. It's just a hold-over from the `colect` methods of formatting data upon which it is based.)

Author(s)

Richard Chandler <rchandler@usgs.gov>

References

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

Chandler, R. B, J. A. Royle, and D. I. King. 2011. Inference about density and temporary emigration in unmarked populations. *Ecology* 92:1429-1435.

See Also

[distsamp](#)

Examples

```
# Simulate some line-transect data

set.seed(36837)

R <- 50 # number of transects
T <- 5  # number of replicates
```

```

strip.width <- 50
transect.length <- 100
breaks <- seq(0, 50, by=10)

lambda <- 5 # Abundance
phi <- 0.6 # Availability
sigma <- 30 # Half-normal shape parameter

J <- length(breaks)-1
y <- array(0, c(R, J, T))
for(i in 1:R) {
  M <- rpois(1, lambda) # Individuals within the 1-ha strip
  for(t in 1:T) {
    # Distances from point
    d <- runif(M, 0, strip.width)
    # Detection process
    if(length(d)) {
      cp <- phi*exp(-d^2 / (2 * sigma^2)) # half-normal w/ g(0)<1
      d <- d[rbinom(length(d), 1, cp) == 1]
      y[i,,t] <- table(cut(d, breaks, include.lowest=TRUE))
    }
  }
}
y <- matrix(y, nrow=R) # convert array to matrix

# Organize data
umf <- unmarkedFrameGDS(y = y, survey="line", unitsIn="m",
  dist.breaks=breaks, tlength=rep(transect.length, R), numPrimary=T)
summary(umf)

# Fit the model
m1 <- gdistsamp(~1, ~1, ~1, umf, output="density")
summary(m1)

backTransform(m1, type="lambda")
backTransform(m1, type="phi")
backTransform(m1, type="det")

# Empirical Bayes estimates of abundance at each site
re <- ranef(m1)
plot(re, layout=c(10,5), xlim=c(-1, 20))

```

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	<i>Generalized multinomial-mixture model</i>
----------	--

Description

A three level hierarchical model for designs involving repeated counts that yield multinomial outcomes. Possible data collection methods include repeated 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", 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
se	Logical. Should standard errors be calculated?
...	Additional arguments to optim, such as lower and upper bounds

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 modeled as binomial: $N_{it} \sim Binomial(M_i, \phi_{ij})$.

The detection process is modeled as multinomial: $\mathbf{y}_{it} \sim Multinomial(N_{it}, \pi_{it})$, where π_{ijt} 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 probabilities 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 <rchandler@usgs.gov> 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.

Chandler, R. B., J. A. Royle, and D. I. King. In Press. Inference about density and temporary emigration in unmarked populations. *Ecology*

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, K=30))

backTransform(m1, type="lambda")      # Individuals per plot
backTransform(m1, type="phi")        # Probability of being available
(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))

# Empirical Bayes estimates of super-population size
re <- ranef(m1)
plot(re, layout=c(5,5), xlim=c(-1,20), subset=site%in%1:25)

```

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 whose 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 Avian 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(masspcru)
```

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	<i>Model selection results from an unmarkedFitList</i>
--------	--

Description

Model selection results from an unmarkedFitList

Arguments

object	an object of class "unmarkedFitList" created by the function <code>fitList</code> .
nullmod	optional character naming which model in the <code>fitList</code> 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. <code>Converge</code> is optim convergence code. <code>CondNum</code> is model condition number. <code>n</code> is the number of sites. <code>delta</code> is delta AIC. <code>cumltvWt</code> is cumulative AIC weight. <code>Rsq</code> is Nagelkerke's (1991) R-squared index, which is only returned when the <code>nullmod</code> 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@usgs.gov>

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)

f1 <- fitList(Null=fm1, A.=fm2, .A=fm3)
f1

ms <- modSel(f1, nullmod="Null")
ms

coef(ms)                # Estimates only
SE(ms)                  # Standard errors only
(toExport <- as(ms, "data.frame")) # Everything

```

multinomPois

Multinomial-Poisson Mixtures Model

Description

Fit the multinomial-Poisson mixture model to data collected using survey methods such as removal sampling or double observer sampling.

Usage

```

multinomPois(formula, data, starts, method = "BFGS",
  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 <code>optim</code> .
se	logical specifying whether or not to compute standard errors.
...	Additional arguments to <code>optim</code> , such as lower and upper bounds

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 and removal sampling can be analyzed with this function by specifying the proper multinomial cell probabilities. 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. (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](#), [unmarkedFrameMPois](#)

Examples

```
# Simulate independent double observer data
nSites <- 50
lambda <- 10
p1 <- 0.5
p2 <- 0.3
cp <- c(p1*(1-p2), p2*(1-p1), p1*p2)
set.seed(9023)
N <- rpois(nSites, lambda)
y <- matrix(NA, nSites, 3)
for(i in 1:nSites) {
  y[i,] <- rmultinom(1, N[i], c(cp, 1-sum(cp)))[1:3]
}

# Fit model
observer <- matrix(c('A','B'), nSites, 2, byrow=TRUE)
umf <- unmarkedFrameMPois(y=y, obsCovs=list(observer=observer),
  type="double")
fm <- multinomPois(~observer-1 ~1, umf)
```

```

# Estimates of fixed effects
e <- coef(fm)
exp(e[1])
plogis(e[2:3])

# Estimates of random effects
re <- ranef(fm, K=20)
#ltheme <- canonical.theme(color = FALSE)
#lattice.options(default.theme = ltheme)
plot(re, layout=c(10,5))

## Real data
data(ovendata)
ovenFrame <- unmarkedFrameMPois(ovendata.list$data,
  siteCovs=as.data.frame(scale(ovendata.list$covariates[, -1])),
  type = "removal")
(fm1 <- multinomPois(~ 1 ~ ufp + trba, ovenFrame))

# Detection probability for a single pass
backTransform(fm1, type="det")

# Detection probability after 4 removal passes
rowSums(getP(fm1))

# Empirical Bayes estimates of abundance at first 25 sites
# Very low uncertainty because p is very high
plot(ranef(fm1, K=10), layout=c(10,7), xlim=c(-1, 10))

```

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 et al. (2002) Occupancy Model

Description

This function fits the single season occupancy model of MacKenzie et al (2002).

Usage

```
occu(formula, data, knownOcc=numeric(0), starts, method="BFGS",
      se=TRUE, engine=c("C", "R"), ...)
```

Arguments

formula	Double right-hand side formula describing covariates of detection and occupancy in that order.
data	An <code>unmarkedFrameOccu</code> object
knownOcc	Vector of sites that are known to be occupied. These should be supplied as row numbers of the y matrix, eg, c(3,8) if sites 3 and 8 were known to be occupied a priori.
starts	Vector of parameter starting values.
method	Optimization method used by <code>optim</code> .
se	Logical specifying whether or not to compute standard errors.
engine	Either "C" or "R" to use fast C++ code or native R code during the optimization.
...	Additional arguments to <code>optim</code> , such as lower and upper bounds

Details

See `unmarkedFrame` and `unmarkedFrameOccu` for a description of how to supply data to the data 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

$$z_i \sim \text{Bernoulli}(\psi_i)$$

The observation process is modeled as

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

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. 2002. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. *Ecology* 83: 2248-2255.

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

Royle, J. A. and R. Dorazio. 2008. *Hierarchical Modeling and Inference in Ecology*. Academic Press.

See Also

[unmarked](#), [unmarkedFrameOccu](#), [modSel](#), [parboot](#)

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
(bt1c <- backTransform(lc))
confint(bt1c, level = 0.9)

# Empirical Bayes estimates of proportion of sites occupied with 95% CI
# No uncertainty because p is so high!
re <- ranef(fm)
c("PAO"=sum(bup(re, stat="mode")), colSums(confint(re)))
```

occuRN

Fit the occupancy model of Royle and Nichols (2003)

Description

Fit the occupancy model of Royle and Nichols (2003)

Usage

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

Arguments

formula	double right-hand side formula describing covariates of detection and occupancy in that order.
data	Object of class unmarkedFrameOccu supplying data to the model.

K	the upper summation index used to numerically integrate out the latent abundance. This should be set high enough so that it does not affect the parameter estimates. Computation time will increase with K.
starts	initial values for the optimization.
method	Optimization method used by <code>optim</code> .
se	logical specifying whether or not to compute standard errors.
...	Additional arguments to <code>optim</code> , such as lower and upper bounds

Details

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

The latent abundance of site i is modelled as Poisson:

$$N_i \sim \text{Poisson}(\lambda_i)$$

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

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

.

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

```
## Not run:

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

# Empirical Bayes estimates of abundance at each site
re <- ranef(fm.wood.rn)
plot(re)

## End(Not run)
```

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@usgs.gov>

See Also

[ranef](#)

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 three fit-statistics.
fitstats <- function(fm) {
  observed <- getY(fm@data)
  expected <- fitted(fm)
  resids <- residuals(fm)
  sse <- sum(resids^2)
  chisq <- sum((observed - expected)^2 / expected)
  freeTuke <- sum((sqrt(observed) - sqrt(expected))^2)
  out <- c(SSE=sse, Chisq=chisq, freemanTukey=freeTuke)
  return(out)
}

(pb <- parboot(fm, fitstats, nsim=25, report=1))
plot(pb, main="")

# Finite-sample inference for a derived parameter.
# Population size in sampled area

Nhat <- function(fm) {
  sum(bup(ranef(fm, K=50)))
}

set.seed(345)
(pb.N <- parboot(fm, Nhat, nsim=25, report=5))

# Compare to empirical Bayes confidence intervals
colSums(confint(ranef(fm, K=50)))

```

Description

Fit the N-mixture model of Royle (2004)

Usage

```
pcount(formula, data, K, mixture=c("P", "NB", "ZIP"),
       starts, method="BFGS", se=TRUE, engine=c("C", "R"), ...)
```

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. This should be set high enough so that it does not affect the parameter estimates. Note that computation time will increase with K.
mixture	character specifying mixture: either "P" or "NB".
starts	vector of starting values
method	Optimization method used by optim .
se	logical specifying whether or not to compute standard errors.
engine	Either "C" or "R" to use fast C++ code or native R code during the optimization.
...	Additional arguments to optim, such as lower and upper bounds

Details

This function fits N-mixture model of Royle (2004) to spatially replicated count data.

See [unmarkedFramePCount](#) for a description of how to format data for pcount.

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 a Poisson, negative binomial, or zero-inflated Poisson random variable, depending on the setting of the mixture argument, mixture = "P", mixture = "NB", mixture = "ZIP" respectively. For the first two distributions, 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). For the ZIP distribution, the mean is $\lambda_i(1 - \psi)$, where psi is the zero-inflation parameter.

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 and Richard Chandler

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 Avian Abundance from Replicated Counts Using Binomial Mixture Models. *Ecological Applications* 15(4), pp. 1450–1461.

Johnson, N.L., A.W. Kemp, and S. Kotz. (2005) *Univariate Discrete Distributions*, 3rd ed. Wiley.

See Also

[unmarkedFramePCount](#), [pcountOpen](#), [ranef](#), [parboot](#)

Examples

```
# Simulate data
set.seed(35)
nSites <- 100
nVisits <- 3
x <- rnorm(nSites)           # a covariate
beta0 <- 0
beta1 <- 1
lambda <- exp(beta0 + beta1*x) # expected counts at each site
N <- rpois(nSites, lambda)    # latent abundance
y <- matrix(NA, nSites, nVisits)
p <- c(0.3, 0.6, 0.8)       # detection prob for each visit
for(j in 1:nVisits) {
  y[,j] <- rbinom(nSites, N, p[j])
}

# Organize data
visitMat <- matrix(as.character(1:nVisits), nSites, nVisits, byrow=TRUE)

umf <- unmarkedFramePCount(y=y, siteCovs=data.frame(x=x),
  obsCovs=list(visit=visitMat))
summary(umf)

# Fit a model
fm1 <- pcount(~visit-1 ~ x, umf, K=50)
fm1

plogis(coef(fm1, type="det")) # Should be close to p

# Empirical Bayes estimation of random effects
(fm1re <- ranef(fm1))
plot(fm1re, subset=site %in% 1:25, xlim=c(-1,40))
sum(bup(fm1re))           # Estimated population size
colSums(confint(fm1re)) # and 95% CI
sum(N)                   # Actual population size
```

```
## Not run:

# Real data
data(mallard)
mallardUMF <- unmarkedFramePCount(mallard.y, siteCovs = mallard.site,
obsCovs = mallard.obs)
(fm.mallard <- pcount(~ ivel+ date + I(date^2) ~ length + elev + forest, mallardUMF, K=30))
(fm.mallard.nb <- pcount(~ date + I(date^2) ~ length + elev, mixture = "NB", mallardUMF, K=30))

## End(Not run)
```

pcountOpen

Fit the open N-mixture model of Dail and Madsen

Description

Fit the model of Dail and Madsen (2011), which is a generalized form of the Royle (2004) N-mixture model for open populations.

Usage

```
pcountOpen(lambdaformula, gammaformula, omegaformula, pformula,
  data, mixture = c("P", "NB", "ZIP"), K, dynamics=c("constant", "autoreg",
  "notrend", "trend"), fix=c("none", "gamma", "omega"), starts,
  method = "BFGS", se = TRUE, ...)
```

Arguments

lambdaformula	Right-hand sided formula for initial abundance
gammaformula	Right-hand sided formula for recruitment rate
omegaformula	Right-hand sided formula for apparent survival probability
pformula	Right-hand sided formula for detection probability
data	An object of class <code>unmarkedFramePCO</code> . See details
mixture	character specifying mixture: "P", "NB", or "ZIP" for the Poisson, negative binomial, and zero-inflated Poisson distributions.
K	Integer defining upper bound of discrete integration. This should be higher than the maximum observed count and high enough that it does not affect the parameter estimates. However, the higher the value the slower the computation.
dynamics	Character string describing the type of population dynamics. "constant" indicates that there is no relationship between omega and gamma. "autoreg" is an auto-regressive model in which recruitment is modeled as $\gamma * N[i,t-1]$. "notrend" model gamma as $\lambda * (1 - \omega)$ such that there is no temporal trend. "trend" is a model for exponential growth, $N[i,t] = N[i,t-1] * \gamma$, where gamma in this case is finite rate of increase (normally referred to as lambda).

fix	If "omega", omega is fixed at 1. If "gamma", gamma is fixed at 0.
starts	vector of starting values
method	Optimization method used by optim .
se	logical specifying whether or not to compute standard errors.
...	additional arguments to be passed to optim .

Details

This model generalizes the Royle (2004) N-mixture model by relaxing the closure assumption. The model includes two additional parameters: gamma, the recruitment rate (births and immigrations), and omega, the apparent survival rate (deaths and emigrations). Estimates of population size at each time period can be derived from these parameters, and thus so can trend estimates. Or, trend can be estimated directly using `dynamics="trend"`.

The latent abundance distribution, $f(N|\theta)$ can be set as a Poisson, negative binomial, or zero-inflated Poisson random variable, depending on the setting of the `mixture` argument, `mixture = "P"`, `mixture = "NB"`, `mixture = "ZIP"` respectively. For the first two distributions, 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). For the ZIP distribution, the mean is $\lambda_i(1 - \psi)$, where ψ is the zero-inflation parameter.

The latent abundance state following the initial sampling period arises from a Markovian process in which survivors are modeled as $S_{it} \sim Binomial(N_{it-1}, \omega_{it})$, and recruits follow $G_{it} \sim Poisson(\gamma_{it})$. Alternative population dynamics can be specified using the `dynamics` argument.

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

λ_i and γ_{it} are modeled using the the log link. ω_{it} and p_{ijt} are modeled using the logit link.

Value

An object of class `unmarkedFitPCO`.

Warning

This function can be extremely slow, especially if there are covariates of gamma or omega. Consider testing the timing on a small subset of the data, perhaps with `se=FALSE`. Finding the lowest value of `K` that does not affect estimates will also help with speed.

Note

When gamma or omega are modeled using year-specific covariates, the covariate data for the final year will be ignored; however, they must be supplied.

If the time gap between primary periods is not constant, an M by T matrix of integers should be supplied to `unmarkedFramePCO` using the `primaryPeriod` argument.

Secondary sampling periods are optional, but can greatly improve the precision of the estimates.

Author(s)

Richard Chandler <rchandler@usgs.gov>

References

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

Dail, D. and L. Madsen (2011) Models for Estimating Abundance from Repeated Counts of an Open Metapopulation. *Biometrics*. 67, pp 577-587.

See Also

[pcount](#), [unmarkedFramePCO](#)

Examples

```
## Simulation
## No covariates, constant time intervals between primary periods, and
## no secondary sampling periods

set.seed(3)
M <- 50
T <- 5
lambda <- 4
gamma <- 1.5
omega <- 0.8
p <- 0.7
y <- N <- matrix(NA, M, T)
S <- G <- matrix(NA, M, T-1)
N[,1] <- rpois(M, lambda)
for(t in 1:(T-1)) {
  S[,t] <- rbinom(M, N[,t], omega)
  G[,t] <- rpois(M, gamma)
  N[,t+1] <- S[,t] + G[,t]
}
y[] <- rbinom(M*T, N, p)

# Prepare data
umf <- unmarkedFramePCO(y = y, numPrimary=T)
summary(umf)

# Fit model and backtransform
(m1 <- pcountOpen(~1, ~1, ~1, ~1, umf, K=20)) # Typically, K should be higher

(lam <- coef(backTransform(m1, "lambda")) # or
lam <- exp(coef(m1, type="lambda"))
gam <- exp(coef(m1, type="gamma"))
om <- plogis(coef(m1, type="omega"))
p <- plogis(coef(m1, type="det"))

# Finite sample inference. Abundance at site i, year t
```

```

re <- ranef(m1)
devAskNewPage(TRUE)
plot(re, layout=c(5,5), subset = site %in% 1:25 & year %in% 1:2,
      xlim=c(-1,15))
devAskNewPage(FALSE)

(N.hat1 <- colSums(bup(re)))
CI <- apply(confint(re), c(2,3), sum)

# Expected values of N[i,t]
N.hat2 <- matrix(NA, M, T)
N.hat2[,1] <- lam
for(t in 2:T) {
  N.hat2[,t] <- om*N.hat2[,t-1] + gam
}

rbind(N=colSums(N), N.hat1=N.hat1, N.hat2=colSums(N.hat2))

plot(1:T, N.hat1, ylim=c(0,600))
points(1:T, colSums(N), col="blue", pch=16)
arrows(1:T, CI[1,], 1:T, CI[2,], code=3, length=0.05, angle=90)

```

piFuns

Compute multinomial cell probabilities

Description

Compute the cell probabilities used in the multinomial-Poisson models [multinomPois](#) and [gmult-mix](#).

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 (see the example).

Value

For removalPiFun, a matrix of cell probabilities 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) # if y has 3 columns
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")
})
```

predict-methods

Methods for Function predict in Package 'unmarked'

Description

These methods return predicted values from fitted model objects.

Methods

```
signature(object = "unmarkedFit") Type must be either 'state' or 'det'.
signature(object = "unmarkedFitColExt") Type must be either 'state' or 'det'.
signature(object = "unmarkedFitGMM")
signature(object = "unmarkedFitList")
```

Description

Estimate posterior distributions of the random variables (latent abundance or occurrence) using empirical Bayes methods. These methods return an object storing the posterior distributions of the latent variables at each site, and for each year (primary period) in the case of open population models. See [unmarkedRanef-class](#) for methods used to manipulate the returned object.

Methods

signature(object = "unmarkedFitOccu") Computes the conditional distribution of occurrence given the data and the estimates of the fixed effects, $Pr(z_i = 1 | y_{ij}, \hat{\psi}_i, \hat{p}_{ij})$

signature(object = "unmarkedFitOccuRN") Computes the conditional abundance distribution given the data and the estimates of the fixed effects, $Pr(N_i = k | y_{ij}, \hat{\psi}_i, \hat{r}_{ij}) k = 0, 1, \dots, K$

signature(object = "unmarkedFitPCount") $Pr(N_i = k | y_{ij}, \hat{\lambda}_i, \hat{p}_{ij}) k = 0, 1, \dots, K$

signature(object = "unmarkedFitMPois") $Pr(N_i = k | y_{ij}, \hat{\lambda}_i, \hat{p}_{ij}) k = 0, 1, \dots, K$

signature(object = "unmarkedFitDS") $Pr(N_i = k | y_{i,1:J}, \hat{\lambda}_i, \hat{\sigma}_i) k = 0, 1, \dots, K$

signature(object = "unmarkedFitGMM") $Pr(M_i = k | y_{i,1:J,t}, \hat{\lambda}_i, \hat{\phi}_{it}, \hat{p}_{ijt}) k = 0, 1, \dots, K$

signature(object = "unmarkedFitGDS") $Pr(M_i = k | y_{i,1:J,t}, \hat{\lambda}_i, \hat{\phi}_{it}, \hat{\sigma}_{it}) k = 0, 1, \dots, K$

signature(object = "unmarkedFitColExt") $Pr(z_{it} = 1 | y_{ijt}, \hat{\psi}_i, \hat{\gamma}_{it}, \hat{\epsilon}_{it}, \hat{p}_{ijt})$

signature(object = "unmarkedFitPCO") $Pr(N_{it} = k | y_{ijt}, \hat{\lambda}_i, \hat{\gamma}_{it}, \hat{\omega}_{it}, \hat{p}_{ijt}) k = 0, 1, \dots, K$

Warning

Empirical Bayes methods can underestimate the variance of the posterior distribution because they do not account for uncertainty in the hyperparameters (lambda or psi). However, we have not found this to be the case in a set of limited simulations studies that can be found in the inst/unitTests directory of unmarked. Simulations do, however, indicate that the posterior mode can exhibit slight (3-5 percent) negative bias as a point estimator of site-specific abundance. It appears to be safer to use the posterior mean even though this will not be an integer in general.

Note

From Carlin and Louis (1996): "... the Bayesian approach to inference depends on a prior distribution for the model parameters. This prior can depend on unknown parameters which in turn may follow some second-stage prior. This sequence of parameters and priors constitutes a hierarchical model. The hierarchy must stop at some point, with all remaining prior parameters assumed known. Rather than make this assumption, the basic empirical Bayes approach uses the observed data to estimate these final stage parameters (or to estimate the Bayes rule), and proceeds as in a standard Bayesian analysis."

Author(s)

Richard Chandler <rchandler@usgs.gov>

References

Laird, N.M. and T.A. Louis. 1987. Empirical Bayes confidence intervals based on bootstrap samples. *Journal of the American Statistical Association* 82:739–750.

Carlin, B.P and T.A Louis. 1996. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall/CRC.

Royle, J.A and R.M. Dorazio. 2008. *Hierarchical Modeling and Inference in Ecology*. Academic Press.

See Also

[unmarkedRanef-class](#)

Examples

```
# Simulate data under N-mixture model
set.seed(4564)
R <- 20
J <- 5
N <- rpois(R, 10)
y <- matrix(NA, R, J)
y[] <- rbinom(R*J, N, 0.5)

# Fit model
umf <- unmarkedFramePCount(y=y)
fm <- pcount(~1 ~1, umf, K=50)

# Estimates of conditional abundance distribution at each site
(re <- ranef(fm))
# Best Unbiased Predictors
bup(re, stat="mean")          # Posterior mean
bup(re, stat="mode")         # Posterior mode
confint(re, level=0.9) # 90% CI

# Plots
plot(re, subset=site %in% c(1:10), layout=c(5, 2), xlim=c(-1,20))

# Compare estimates to truth
sum(N)
sum(bup(re))
colSums(confint(re))

# Extract all values in convenient formats
post.df <- as(re, "data.frame")
head(post.df)
post.arr <- as(re, "array")
```

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 couple of models
(fm1 <- multinomPois(~ 1 ~ ufp + trba, ovenFrame))
summary(fm1)

# Apply a bunch of methods to the fitted model

# Look at the different parameter types
names(fm1)
fm1['state']
fm1['det']
```

```

# Coefficients from abundance part of the model
coef(fm1, type='state')

# Variance-covariance matrix
vcov(fm1, type='state')

# Confidence intervals using profiled likelihood
confint(fm1, type='state', method='profile')

# Expected values
fitted(fm1)

# Original data
getData(fm1)

# Detection probabilities
getP(fm1)

# log-likelihood
logLik(fm1)

# Back-transform detection probability to original scale
# backTransform only works on models with no covariates or
#   in conjunction with linearComb (next example)
backTransform(fm1, type='det')

# Predicted abundance at specified covariate values
(lc <- linearComb(fm1, c(Int = 1, ufp = 0, trba = 0), type='state'))
backTransform(lc)

# 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)

# Number of sites in the sample
sampleSize(fm1)

# Fit a new model without covariates
(fmNull <- update(fm1, formula = ~1 ~1))

# Likelihood ratio test
LRT(fm1, fmNull)

```

unmarkedFitList-class *Class "unmarkedFitList"*

Description

Class to hold multiple 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

coef signature(object = "unmarkedFitList"): Extract coefficients

SE signature(object = "unmarkedFitList"): Extract standard errors

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")

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)

f1 <- fitList(Null=fm1, A.=fm2, .A=fm3)
f1

coef(f1)
SE(f1)
```

```
ms <- modSel(f1, nullmod="Null")
ms
```

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	An MxJ matrix of the observed measured data, where M is the number of sites and J is the maximum number of observations per site.
siteCovs	A data.frame of covariates that vary at the site level. This should have M rows and one column per covariate
obsCovs	Either a named list of data.frames of covariates that vary within sites, or a data.frame with MxJ rows in site-major order.
obsToY	optional matrix specifying relationship between observation-level covariates and response matrix
mapInfo	geographic coordinate information. Currently ignored.

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 covariate 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 additional metadata. For example, unmarkedFrameDS is used to organize distance sampling data for the [distsamp](#) function, and it has arguments dist.breaks,

length, 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](#), [unmarkedFrameOccu](#), [unmarkedFramePCount](#), [unmarkedFrameDS](#)

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
```

unmarkedFrameDS	<i>Organize data for the distance sampling model of Royle et al. (2004) fit by <code>distsamp</code></i>
-----------------	--

Description

Organizes count data along with the covariates and metadata. This S4 class is required by the data argument of `distsamp`

Usage

```
unmarkedFrameDS(y, siteCovs=NULL, dist.breaks, tlength, survey,
  unitsIn, mapInfo)
```

Arguments

<code>y</code>	An RxJ matrix of count data, where R is the number of sites (transects) and J is the number of distance classes.
<code>siteCovs</code>	A <code>data.frame</code> of covariates that vary at the site level. This should have R rows and one column per covariate
<code>dist.breaks</code>	vector of distance cut-points delimiting the distance classes. It must be of length J+1.
<code>tlength</code>	A vector of length R containing the transect lengths. This is ignored when <code>survey="point"</code> .
<code>survey</code>	Either "point" or "line" for point- and line-transects.
<code>unitsIn</code>	Either "m" or "km" defining the measurement units for <i>both</i> <code>dist.breaks</code> and <code>tlength</code> .
<code>mapInfo</code>	Currently ignored

Details

`unmarkedFrameDS` is the S4 class that holds data to be passed to the `distsamp` model-fitting function.

Value

an object of class `unmarkedFrameDS`

Note

If you have continuous distance data, they must be "binned" into discrete distance classes, which are delimited by `dist.breaks`.

References

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

See Also

[unmarkedFrame-class](#), [unmarkedFrame](#), [distsamp](#)

Examples

```
# Fake data
R <- 4 # number of sites
J <- 3 # number of distance classes

db <- c(0, 10, 20, 30) # distance break points

y <- matrix(c(
  5,4,3, # 5 detections in 0-10 distance class at this transect
  0,0,0,
  2,1,1,
  1,1,0), nrow=R, ncol=J, byrow=TRUE)
y

site.covs <- data.frame(x1=1:4, x2=factor(c('A','B','A','B')))
site.covs

umf <- unmarkedFrameDS(y=y, siteCovs=site.covs, dist.breaks=db, survey="point",
  unitsIn="m") # organize data
umf           # look at data
summary(umf)  # summarize
fm <- distsamp(~1 ~1, umf) # fit a model
```

unmarkedFrameMPois	<i>Organize data for the multinomial-Poisson mixture model of Royle (2004) fit by multinomPois</i>
--------------------	--

Description

Organizes count data along with the covariates. This S4 class is required by the data argument of [multinomPois](#)

Usage

```
unmarkedFrameMPois(y, siteCovs=NULL, obsCovs=NULL, type, obsToY,
  mapInfo, piFun)
```

Arguments

y An RxJ matrix of count data, where R is the number of sites (transects) and J is the maximum number of observations per site.

siteCovs	A <code>data.frame</code> of covariates that vary at the site level. This should have R rows and one column per covariate
obsCovs	Either a named list of RxJ <code>data.frames</code> or a <code>data.frame</code> with RxJ rows and one column per covariate. For the latter format, the covariates should be in site-major order.
type	Either "removal" or "double" for removal sampling or double observer sampling. If this argument not specified, the user must provide an obsToY matrix. See details.
obsToY	A matrix describing the relationship between obsCovs and y. This is necessary because under some sampling designs the dimensions of y do not equal the dimensions of each observation level covariate. For example, in double observer sampling there are 3 observations (seen only by observer A, detected only by observer B, and detected by both), but each observation-level covariate can only have 2 columns, one for each observer. This matrix is created automatically if type is either "removal" or "double".
mapInfo	Currently ignored
piFun	Function used to compute the multinomial cell probabilities from a matrix of detection probabilities. This is created automatically if type is either "removal" or "double".

Details

unmarkedFrameMPois is the S4 class that holds data to be passed to the `multinomPois` model-fitting function.

Value

an object of class unmarkedFrameMPois

References

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

See Also

[unmarkedFrame-class](#), [unmarkedFrame](#), [multinomPois](#), [piFuns](#)

Examples

```
# Fake double observer data
R <- 4 # number of sites
J <- 2 # number of observers

y <- matrix(c(
  1,0,3,
  0,0,0,
  2,0,1,
```

```

      0,0,2), nrow=R, ncol=J+1, byrow=TRUE)
y

site.covs <- data.frame(x1=1:4, x2=factor(c('A','B','A','B')))
site.covs

obs.covs <- list(
  x3 = matrix(c(
    -1,0,
    -2,0,
    -3,1,
    0,0),
    nrow=R, ncol=J, byrow=TRUE),
  x4 = matrix(c(
    'a','b',
    'a','b',
    'a','b',
    'a','b'),
    nrow=R, ncol=J, byrow=TRUE))
obs.covs

# Create unmarkedFrame
umf <- unmarkedFrameMPois(y=y, siteCovs=site.covs, obsCovs=obs.covs,
  type="double")

# The above is the same as:
o2y <- matrix(1, 2, 3)
pifun <- function(p)
{
  M <- nrow(p)
  pi <- matrix(NA, M, 3)
  pi[, 1] <- p[, 1] * (1 - p[, 2])
  pi[, 2] <- p[, 2] * (1 - p[, 1])
  pi[, 3] <- p[, 1] * p[, 2]
  return(pi)
}

umf <- unmarkedFrameMPois(y=y, siteCovs=site.covs, obsCovs=obs.covs,
  obsToY=o2y, piFun="pifun")

# Fit a model
fm <- multinomPois(~1 ~1, umf)

```

Description

Organizes detection, non-detection data along with the covariates. This S4 class is required by the data argument of [occu](#) and [occuRN](#)

Usage

```
unmarkedFrameOccu(y, siteCovs=NULL, obsCovs=NULL, mapInfo)
```

Arguments

<code>y</code>	An RxJ matrix of the detection, non-detection data, where R is the number of sites, J is the maximum number of sampling periods per site.
<code>siteCovs</code>	A data.frame of covariates that vary at the site level. This should have M rows and one column per covariate
<code>obsCovs</code>	Either a named list of data.frames of covariates that vary within sites, or a data.frame with RxJ rows in site-major order.
<code>mapInfo</code>	Currently ignored

Details

`unmarkedFrameOccu` is the S4 class that holds data to be passed to the [occu](#) and [occuRN](#) model-fitting function.

Value

an object of class `unmarkedFrameOccu`

See Also

[unmarkedFrame-class](#), [unmarkedFrame](#), [occu](#), [occuRN](#)

Examples

```
# Fake data
R <- 4 # number of sites
J <- 3 # number of visits
y <- matrix(c(
  1,1,0,
  0,0,0,
  1,1,1,
  1,0,1), nrow=R, ncol=J, byrow=TRUE)
y

site.covs <- data.frame(x1=1:4, x2=factor(c('A','B','A','B')))
site.covs

obs.covs <- list(
  x3 = matrix(c(
    -1,0,1,
```

```

      -2,0,0,
      -3,1,0,
      0,0,0), nrow=R, ncol=J, byrow=TRUE),
x4 = matrix(c(
  'a','b','c',
  'd','b','a',
  'a','a','c',
  'a','b','a'), nrow=R, ncol=J, byrow=TRUE))
obs.covs

umf <- unmarkedFrameOccu(y=y, siteCovs=site.covs,
  obsCovs=obs.covs) # organize data
umf # look at data
summary(umf) # summarize
fm <- occu(~1 ~1, umf) # fit a model

```

unmarkedFramePCO	<i>Create an object of class unmarkedFramePCO that contains data used by pcountOpen.</i>
------------------	--

Description

Organizes repeated count data along with the covariates and possibly the dates on which each survey was conducted. This S4 class is required by the data argument of [pcountOpen](#)

Usage

```
unmarkedFramePCO(y, siteCovs=NULL, obsCovs=NULL, yearlySiteCovs, mapInfo,
  numPrimary, primaryPeriod)
```

Arguments

y	An MxJT matrix of the repeated count data, where M is the number of sites, J is the maximum number of secondary sampling periods per site and T is the maximum number of primary sampling periods per site.
siteCovs	A data.frame of covariates that vary at the site level. This should have M rows and one column per covariate
obsCovs	Either a named list of data.frames of covariates that vary within sites, or a data.frame with MxJT rows in site-major order.
yearlySiteCovs	Either a named list of MxT data.frames , or a site-major data.frame with MT rows and 1 column per covariate.
mapInfo	Currently ignored
numPrimary	Maximum number of observed primary periods for each site
primaryPeriod	matrix of integers indicating the primary period of each survey.

Details

unmarkedFramePCO is the S4 class that holds data to be passed to the `pcountOpen` model-fitting function.

The unmarkedFramePCO class is similar to the unmarkedFramePCount class except that it contains the dates for each survey, which needs to be supplied .

Value

an object of class unmarkedFramePCO

See Also

[unmarkedFrame-class](#), [unmarkedFrame](#), [pcountOpen](#)

Examples

```
# Repeated count data with 5 primary periods and
# no secondary sampling periods (ie J==1)
y1 <- matrix(c(
  0, 2, 3, 2, 0,
  2, 2, 3, 1, 1,
  1, 1, 0, 0, 3,
  0, 0, 0, 0, 0), nrow=4, ncol=5, byrow=TRUE)

# Site-specific covariates
sc1 <- data.frame(x1 = 1:4, x2 = c('A','A','B','B'))

# Observation-specific covariates
oc1 <- list(
  x3 = matrix(1:5, nrow=4, ncol=5, byrow=TRUE),
  x4 = matrix(letters[1:5], nrow=4, ncol=5, byrow=TRUE))

# Primary periods of surveys
primaryPeriod1 <- matrix(as.integer(c(
  1, 2, 5, 7, 8,
  1, 2, 3, 4, 5,
  1, 2, 4, 5, 6,
  1, 3, 5, 6, 7)), nrow=4, ncol=5, byrow=TRUE)

# Create the unmarkedFrame
umf1 <- unmarkedFramePCO(y=y1, siteCovs=sc1, obsCovs=oc1, numPrimary=5,
  primaryPeriod=primaryPeriod1)

# Take a look
umf1
summary(umf1)
```

```

# Repeated count data with 4 primary periods and
# no 2 secondary sampling periods (ie J=2)
y2 <- matrix(c(
  0,0, 2,2, 3,2, 2,2,
  2,2, 2,1, 3,2, 1,1,
  1,0, 1,1, 0,0, 0,0,
  0,0, 0,0, 0,0, 0,0), nrow=4, ncol=8, byrow=TRUE)

# Site-specific covariates
sc2 <- data.frame(x1 = 1:4, x2 = c('A','A','B','B'))

# Observation-specific covariates
oc2 <- list(
  x3 = matrix(1:8, nrow=4, ncol=8, byrow=TRUE),
  x4 = matrix(letters[1:8], nrow=4, ncol=8, byrow=TRUE))

# Yearly-site covariates
ysc <- list(
  x5 = matrix(c(
    1,2,3,4,
    1,2,3,4,
    1,2,3,4,
    1,2,3,4), nrow=4, ncol=4, byrow=TRUE))

# Primary periods of surveys
primaryPeriod2 <- matrix(as.integer(c(
  1,2,5,7,
  1,2,3,4,
  1,2,4,5,
  1,3,5,6)), nrow=4, ncol=4, byrow=TRUE)

# Create the unmarkedFrame
umf2 <- unmarkedFramePCO(y=y2, siteCovs=sc2, obsCovs=oc2,
  yearlySiteCovs=ysc,
  numPrimary=4, primaryPeriod=primaryPeriod2)

# Take a look
umf2
summary(umf2)

```

Description

Organizes repeated count data along with the covariates. This S4 class is required by the data argument of [pcount](#)

Usage

```
unmarkedFramePCount(y, siteCovs=NULL, obsCovs=NULL, mapInfo)
```

Arguments

<code>y</code>	An RxJ matrix of the repeated count data, where R is the number of sites, J is the maximum number of sampling periods per site.
<code>siteCovs</code>	A data.frame of covariates that vary at the site level. This should have M rows and one column per covariate
<code>obsCovs</code>	Either a named list of data.frames of covariates that vary within sites, or a data.frame with RxJ rows in site-major order.
<code>mapInfo</code>	Currently ignored

Details

`unmarkedFramePCount` is the S4 class that holds data to be passed to the [pcount](#) model-fitting function.

Value

an object of class `unmarkedFramePCount`

See Also

[unmarkedFrame-class](#), [unmarkedFrame](#), [pcount](#)

Examples

```
# Fake data
R <- 4 # number of sites
J <- 3 # number of visits
y <- matrix(c(
  1,2,0,
  0,0,0,
  1,1,1,
  2,2,1), nrow=R, ncol=J, byrow=TRUE)
y

site.covs <- data.frame(x1=1:4, x2=factor(c('A','B','A','B')))
site.covs

obs.covs <- list(
  x3 = matrix(c(
    -1,0,1,
```

```

      -2,0,0,
      -3,1,0,
      0,0,0), nrow=R, ncol=J, byrow=TRUE),
x4 = matrix(c(
  'a','b','c',
  'd','b','a',
  'a','a','c',
  'a','b','a'), nrow=R, ncol=J, byrow=TRUE))
obs.covs

umf <- unmarkedFramePCount(y=y, siteCovs=site.covs,
  obsCovs=obs.covs) # organize data
umf # take a l
summary(umf) # summarize data
fm <- pcount(~1 ~1, umf, K=10) # fit a model

```

unmarkedMultFrame *Create an unmarkedMultiFrame or an unmarkedFrameGMM.*

Description

These functions construct 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)
unmarkedFrameGDS(y, siteCovs, numPrimary, yearlySiteCovs, dist.breaks,
  survey, unitsIn, tlength)

```

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 not be specified for other types of survey designs.
obsToY	A matrix specifying relationship between observation-level covariates and response matrix

<code>piFun</code>	A function converting an MxJ matrix of detection probabilities into an MxJ matrix of multinomial cell probabilities.
<code>dist.breaks</code>	see unmarkedFrameDS
<code>survey</code>	see unmarkedFrameDS
<code>unitsIn</code>	see unmarkedFrameDS
<code>tlength</code>	see unmarkedFrameDS

Details

unmarkedMultFrame objects are used by [colext](#).

unmarkedFrameGMM objects are used by [gmultmix](#).

unmarkedFrameGDS objects are used by [gdistsamp](#).

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 and unmarkedFrameGDS are superclasses of unmarkedMultFrame containing information on the survey design used that resulted in multinomial outcomes. For unmarkedFrameGMM and 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 appropriate `obsToY` matrix and [piFuns](#). For all other situations, the `type` argument of unmarkedFrameGMM 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](#).

unmarkedFrameGMM and unmarkedFrameGDS objects can be created from an unmarkedMultFrame using the "as" conversion method. See examples.

Value

an unmarkedMultFrame or unmarkedFrameGMM object

Note

Data used with [colext](#), [gmultmix](#), and [gdistsamp](#) may be collected during a single year, so `yearlySiteCovs` may be a misnomer in some cases.

See Also

[formatMult](#)

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=list(year=years),
  numPrimary=T)

umfGMM1 <- unmarkedFrameGMM(y=y,
  siteCovs = data.frame(site=site),
  obsCovs=list(occasion=occasions),
  yearlySiteCovs=data.frame(year=c(t(years))),
  # or: yearlySiteCovs=list(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)
```

```
unmarkedRanef-class   Class "unmarkedRanef"
```

Description

Stores the estimated posterior distributions of the latent abundance or occurrence variables.

Objects from the Class

Objects can be created by calls of the form [ranef](#).

Slots

post: An [array](#) with nSites rows and Nmax (K+1) columns and nPrimaryPeriod slices

Methods

bup signature(object = "unmarkedRanef"): Extract the Best Unbiased Predictors (BUPs) of the latent variables (abundance or occurrence state). Either the posterior mean or median can be requested using the `stat` argument.

confint signature(object = "unmarkedRanef"): Compute confidence intervals.

plot signature(x = "unmarkedRanef", y = "missing"): Plot the posteriors using [xyplot](#)

show signature(object = "unmarkedRanef"): Display the modes and confidence intervals

Warnings

Empirical Bayes methods can underestimate the variance of the posterior distribution because they do not account for uncertainty in the hyperparameters (λ or ψ). However, we have not found this to be the case in a set of limited simulations studies that can be found in the `inst/unitTests` directory of `unmarked`. Simulations do, however, indicate that the posterior mode can exhibit slight (3-5 percent) negatively bias as a point estimator of site-specific abundance. It appears to be safer to use the posterior mean even though this will not be an integer in general.

References

- Laird, N.M. and T.A. Louis. 1987. Empirical Bayes confidence intervals based on bootstrap samples. *Journal of the American Statistical Association* 82:739–750.
- Carlin, B.P and T.A Louis. 1996. *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall/CRC.
- Royle, J.A and R.M. Dorazio. 2008. *Hierarchical Modeling and Inference in Ecology*. Academic Press.

See Also[ranef](#)**Examples**

```
showClass("unmarkedRanef")
```

 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 'unmarkedFrame,list,missing,missing'
x[i, j]
## S4 method for signature 'unmarkedMultFrame,missing,numeric,missing'
x[i, j]
## S4 method for signature 'unmarkedMultFrame,numeric,missing,missing'
x[i, j]
## S4 method for signature 'unmarkedFrameGMM,numeric,missing,missing'
x[i, j]
## S4 method for signature 'unmarkedFrameGDS,numeric,missing,missing'
x[i, j]
## S4 method for signature 'unmarkedFramePCO,numeric,missing,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.
- x = "unmarkedMultFrame", i = "numeric", j = "missing", drop = "missing"** Extract rows (sites) from an unmarkedMultFrame
- x = "unmarkedGMM", i = "numeric", j = "missing", drop = "missing"** Extract rows (sites) from an unmarkedFrameGMM object
- x = "unmarkedGDS", i = "numeric", j = "missing", drop = "missing"** Extract rows (sites) from an unmarkedFrameGDS object
- x = "unmarkedPCO", i = "numeric", j = "missing", drop = "missing"** Extract rows (sites) from an unmarkedFramePCO object

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

*Topic **classes**

- unmarkedEstimate-class, 59
- unmarkedEstimateList-class, 60
- unmarkedFit-class, 60
- unmarkedFitList-class, 64
- unmarkedFrame-class, 67
- unmarkedRanef-class, 81

*Topic **datasets**

- birds, 8
- frogs, 24
- gf, 28
- linetran, 33
- mallard, 34
- masspcru, 35
- ovendata, 44
- pointtran, 53

*Topic **methods**

- [–methods, 82
- backTransform-methods, 7
- coef-methods, 8
- confint-methods, 12
- fitted-methods, 19
- getP-methods, 27
- linearComb-methods, 33
- nonparboot-methods, 39
- predict-methods, 54
- ranef-methods, 55
- SE-methods, 57
- simulate-methods, 58
- vcov-methods, 82

*Topic **models**

- colext, 9
- distsamp, 15
- gdistsamp, 25
- multinomPois, 37
- occu, 40
- occuRN, 42
- pcount, 46
- pcountOpen, 49

*Topic **model**

- gmultmix, 29

*Topic **package**

- unmarked-package, 3

*Topic **utilities**

- csvToUMF, 13
- imputeMissing, 31

- [,unmarkedEstimateList,ANY,ANY,ANY-method
([–methods), 82

- [,unmarkedFit,ANY,ANY,ANY-method
([–methods), 82

- [,unmarkedFrame,list,missing,missing-method
([–methods), 82

- [,unmarkedFrame,missing,numeric,missing-method
([–methods), 82

- [,unmarkedFrame,numeric,missing,missing-method
([–methods), 82

- [,unmarkedFrame,numeric,numeric,missing-method
([–methods), 82

- [,unmarkedFrameGDS,numeric,missing,missing-method
([–methods), 82

- [,unmarkedFrameGMM,numeric,missing,missing-method
([–methods), 82

- [,unmarkedFramePCO,numeric,missing,missing-method
([–methods), 82

- [,unmarkedMultFrame,missing,numeric,missing-method
([–methods), 82

- [,unmarkedMultFrame,numeric,missing,missing-method
([–methods), 82

- [–methods, 82

array, 81

backTransform, 39

backTransform (backTransform-methods), 7
backTransform,unmarkedEstimate-method
(backTransform-methods), 7

backTransform,unmarkedFit-method
(backTransform-methods), 7

- backTransform, unmarkedLinComb-method
(backTransform-methods), 7
- backTransform-methods, 7
- birds, 8
- bup (unmarkedRanef-class), 81
- bup, unmarkedRanef-method
(unmarkedRanef-class), 81

- catbird (birds), 8
- coef, linCombOrBackTrans-method
(coef-methods), 8
- coef, unmarkedEstimate-method
(coef-methods), 8
- coef, unmarkedFit-method (coef-methods),
8
- coef, unmarkedFitList-method
(unmarkedFitList-class), 64
- coef, unmarkedModSel-method (modSel), 36
- coef-methods, 8
- colext, 3, 4, 9, 20, 58, 79
- confint, unmarkedBackTrans-method
(confint-methods), 12
- confint, unmarkedEstimate-method
(confint-methods), 12
- confint, unmarkedFit-method
(confint-methods), 12
- confint, unmarkedLinComb-method
(confint-methods), 12
- confint, unmarkedRanef-method
(unmarkedRanef-class), 81
- confint-methods, 12
- coordinates (unmarkedFrame-class), 67
- coordinates, unmarkedFrame-method
(unmarkedFrame-class), 67
- coords (unmarkedFrame-class), 67
- csvToUMF, 4, 13, 24

- data.frame, 65, 69, 71, 73, 74, 77
- detFuns, 14, 17
- distsamp, 3, 4, 14, 15, 20, 21, 26, 32, 57,
58, 61, 65, 69, 70
- doublePiFun, 29, 38
- doublePiFun (piFuns), 52
- drexp (detFuns), 14
- drhaz (detFuns), 14
- drhn (detFuns), 14
- dxexp (detFuns), 14
- dxhaz (detFuns), 14
- dxhn (detFuns), 14

- fitList, 17, 18, 36, 64
- fitted, unmarkedFit-method
(fitted-methods), 19
- fitted, unmarkedFitColExt-method
(fitted-methods), 19
- fitted, unmarkedFitDS-method
(fitted-methods), 19
- fitted, unmarkedFitGMM-method
(fitted-methods), 19
- fitted, unmarkedFitOccu-method
(fitted-methods), 19
- fitted, unmarkedFitOccuRN-method
(fitted-methods), 19
- fitted, unmarkedFitPCO-method
(fitted-methods), 19
- fitted, unmarkedFitPCount-method
(fitted-methods), 19
- fitted-methods, 19
- formatDistData, 17, 20
- formatLong (formatWideLong), 23
- formatMult, 10, 22, 79
- formatWide (formatWideLong), 23
- formatWideLong, 23
- formula, 41
- frog2001pcru (frogs), 24
- frog2001pfer (frogs), 24
- frogs, 24

- gdistsamp, 4, 17, 20, 21, 25, 79
- getData (unmarkedFit-class), 60
- getData, unmarkedFit-method
(unmarkedFit-class), 60
- getP (getP-methods), 27
- getP, unmarkedFit-method (getP-methods),
27
- getP, unmarkedFitColExt-method
(getP-methods), 27
- getP, unmarkedFitDS-method
(getP-methods), 27
- getP, unmarkedFitGDS-method
(getP-methods), 27
- getP, unmarkedFitGMM-method
(getP-methods), 27
- getP, unmarkedFitMPois-method
(getP-methods), 27
- getP, unmarkedFitPCO-method
(getP-methods), 27
- getP-methods, 27
- getY (unmarkedFrame-class), 67

- getY, unmarkedFrame-method
(unmarkedFrame-class), 67
- gf, 28
- gmultmix, 4, 29, 52, 79
- grep (detFuns), 14
- grhaz (detFuns), 14
- grhn (detFuns), 14
- gexp (detFuns), 14
- gxhaz (detFuns), 14
- gxhn (detFuns), 14
- head, unmarkedFrame-method
(unmarkedFrame-class), 67
- hessian (unmarkedFit-class), 60
- hessian, unmarkedFit-method
(unmarkedFit-class), 60
- hist, unmarkedFitDS-method
(unmarkedFit-class), 60
- hist, unmarkedFrameDS-method
(unmarkedFrame-class), 67
- imputeMissing, 31
- integrate, 25
- lambda2psi, 32
- linearComb, 39
- linearComb (linearComb-methods), 33
- linearComb, unmarkedEstimate, matrixOrVector-method
(linearComb-methods), 33
- linearComb, unmarkedFit, matrixOrVector-method
(linearComb-methods), 33
- linearComb-methods, 82
- linearComb-methods, 33
- linetran, 33
- logLik (unmarkedFit-class), 60
- logLik, unmarkedFit-method
(unmarkedFit-class), 60
- LRT (unmarkedFit-class), 60
- LRT, unmarkedFit, unmarkedFit-method
(unmarkedFit-class), 60
- mallard, 34
- mapInfo (unmarkedFrame-class), 67
- masspcru, 35
- mle (unmarkedFit-class), 60
- mle, unmarkedFit-method
(unmarkedFit-class), 60
- modSel, 36, 42
- modSel, unmarkedFitList-method
(unmarkedFitList-class), 64
- modSel-methods (modSel), 36
- multinomPois, 4, 30, 32, 37, 52, 58, 65, 70,
71, 79
- names, unmarkedEstimateList-method
(unmarkedEstimateList-class),
60
- names, unmarkedFit-method
(unmarkedFit-class), 60
- nllFun (unmarkedFit-class), 60
- nllFun, unmarkedFit-method
(unmarkedFit-class), 60
- nonparboot, 10
- nonparboot (nonparboot-methods), 39
- nonparboot, unmarkedFit-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitColExt-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitDS-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitGDS-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitGMM-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitMPois-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitOccu-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitOccuRN-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitPCO-method
(nonparboot-methods), 39
- nonparboot, unmarkedFitPCount-method
(nonparboot-methods), 39
- nonparboot-methods, 39
- numSites (unmarkedFrame-class), 67
- numSites, unmarkedFrame-method
(unmarkedFrame-class), 67
- numY (unmarkedFrame-class), 67
- numY, unmarkedFrame-method
(unmarkedFrame-class), 67
- obsCovs (unmarkedFrame-class), 67
- obsCovs, unmarkedFrame-method
(unmarkedFrame-class), 67
- obsCovs<- (unmarkedFrame-class), 67
- obsCovs<- , unmarkedFrame-method
(unmarkedFrame-class), 67
- obsNum (unmarkedFrame-class), 67

- obsNum, unmarkedFrame-method
(unmarkedFrame-class), 67
- obsToY (unmarkedFrame-class), 67
- obsToY, unmarkedFrame-method
(unmarkedFrame-class), 67
- obsToY<- (unmarkedFrame-class), 67
- obsToY<-, unmarkedFrame-method
(unmarkedFrame-class), 67
- occu, 3, 4, 20, 40, 58, 73
- occuRN, 3, 4, 20, 42, 58, 73
- optim, 9, 16, 25, 29, 37, 41, 43, 47, 50, 61
- ovendata, 44
- parboot, 17, 42, 45, 48, 59
- parboot, unmarkedFit-method
(unmarkedFit-class), 60
- pcount, 3, 4, 20, 32, 46, 51, 58, 77
- pcountOpen, 4, 48, 49, 74, 75
- pcru.bin (frogs), 24
- pcru.data (frogs), 24
- pcru.y (frogs), 24
- pfer.bin (frogs), 24
- pfer.data (frogs), 24
- pfer.y (frogs), 24
- piFuns, 30, 38, 52, 71, 79
- plot, parboot, missing-method (parboot),
45
- plot, profile, missing-method
(unmarkedFit-class), 60
- plot, unmarkedFit, missing-method
(unmarkedFit-class), 60
- plot, unmarkedFrame, missing-method
(unmarkedFrame-class), 67
- plot, unmarkedRanef, missing-method
(unmarkedRanef-class), 81
- pointtran, 53
- powerAnalysis (unmarkedFrame-class), 67
- powerAnalysis, formula, unmarkedFramePCount, numeric-method
(unmarkedFrame-class), 67
- predict, 7
- predict, ANY-method (predict-methods), 54
- predict, unmarkedFit-method
(predict-methods), 54
- predict, unmarkedFitColExt-method
(predict-methods), 54
- predict, unmarkedFitGDS-method
(predict-methods), 54
- predict, unmarkedFitGMM-method
(predict-methods), 54
- predict, unmarkedFitList-method
(predict-methods), 54
- predict, unmarkedFitPCO-method
(predict-methods), 54
- predict-methods, 54
- profile, unmarkedFit-method
(unmarkedFit-class), 60
- projected (unmarkedFit-class), 60
- projected, unmarkedFitColExt-method
(unmarkedFit-class), 60
- projection (unmarkedFrame-class), 67
- projection, unmarkedFrame-method
(unmarkedFrame-class), 67
- ranef, 17, 45, 48, 81, 82
- ranef (ranef-methods), 55
- ranef, unmarkedFitColExt-method
(ranef-methods), 55
- ranef, unmarkedFitDS-method
(ranef-methods), 55
- ranef, unmarkedFitGDS-method
(ranef-methods), 55
- ranef, unmarkedFitGMM-method
(ranef-methods), 55
- ranef, unmarkedFitGMMorGDS-method
(ranef-methods), 55
- ranef, unmarkedFitMPois-method
(ranef-methods), 55
- ranef, unmarkedFitOccu-method
(ranef-methods), 55
- ranef, unmarkedFitOccuRN-method
(ranef-methods), 55
- ranef, unmarkedFitPCO-method
(ranef-methods), 55
- ranef, unmarkedFitPCount-method
(ranef-methods), 55
- ranef-methods, 55
- removalPiFun (piFuns), 52
- residuals, unmarkedFit-method
(unmarkedFit-class), 60
- residuals, unmarkedFitOccu-method
(unmarkedFit-class), 60
- residuals, unmarkedFitOccuRN-method
(unmarkedFit-class), 60
- sampleSize (unmarkedFit-class), 60
- sampleSize, unmarkedFit-method
(unmarkedFit-class), 60

- SE (SE-methods), 57
- SE, linCombOrBackTrans-method
(SE-methods), 57
- SE, unmarkedEstimate-method
(SE-methods), 57
- SE, unmarkedFit-method (SE-methods), 57
- SE, unmarkedFitList-method
(unmarkedFitList-class), 64
- SE, unmarkedModSel-method (modSel), 36
- SE-methods, 57
- show, parboot-method (parboot), 45
- show, unmarkedBackTrans-method
(backTransform-methods), 7
- show, unmarkedEstimate-method
(unmarkedEstimate-class), 59
- show, unmarkedEstimateList-method
(unmarkedEstimateList-class),
60
- show, unmarkedFit-method
(unmarkedFit-class), 60
- show, unmarkedFrame-method
(unmarkedFrame-class), 67
- show, unmarkedLinComb-method
(linearComb-methods), 33
- show, unmarkedModSel-method (modSel), 36
- show, unmarkedMultFrame-method
(unmarkedFrame-class), 67
- show, unmarkedRanef-method
(unmarkedRanef-class), 81
- sight2perpdist, 17, 57
- simulate, unmarkedFitColExt-method
(simulate-methods), 58
- simulate, unmarkedFitDS-method
(simulate-methods), 58
- simulate, unmarkedFitGDS-method
(simulate-methods), 58
- simulate, unmarkedFitGMM-method
(simulate-methods), 58
- simulate, unmarkedFitMPois-method
(simulate-methods), 58
- simulate, unmarkedFitOccu-method
(simulate-methods), 58
- simulate, unmarkedFitOccuRN-method
(simulate-methods), 58
- simulate, unmarkedFitPCO-method
(simulate-methods), 58
- simulate, unmarkedFitPCount-method
(simulate-methods), 58
- simulate-methods, 58
- siteCovs (unmarkedFrame-class), 67
- siteCovs, unmarkedFrame-method
(unmarkedFrame-class), 67
- siteCovs<- (unmarkedFrame-class), 67
- siteCovs<- , unmarkedFrame-method
(unmarkedFrame-class), 67
- smoothed (unmarkedFit-class), 60
- smoothed, unmarkedFitColExt-method
(unmarkedFit-class), 60
- SSE, 59
- summary, unmarkedEstimate-method
(unmarkedEstimate-class), 59
- summary, unmarkedEstimateList-method
(unmarkedEstimateList-class),
60
- summary, unmarkedFit-method
(unmarkedFit-class), 60
- summary, unmarkedFitDS-method
(unmarkedFit-class), 60
- summary, unmarkedFitList-method
(unmarkedFitList-class), 64
- summary, unmarkedFrame-method
(unmarkedFrame-class), 67
- summary, unmarkedFrameDS-method
(unmarkedFrame-class), 67
- summary, unmarkedModSel-method (modSel),
36
- summary, unmarkedMultFrame-method
(unmarkedFrame-class), 67
- unmarked, 13, 42
- unmarked (unmarked-package), 3
- unmarked-package, 68
- unmarked-package, 3
- unmarkedEstimate
(unmarkedEstimate-class), 59
- unmarkedEstimate-class, 57, 82
- unmarkedEstimate-class, 59
- unmarkedEstimateList-class, 60
- unmarkedFit, 60, 64, 68
- unmarkedFit (unmarkedFit-class), 60
- unmarkedFit-class, 13, 16, 17, 60
- unmarkedFit-class, 60
- unmarkedFitDS-class
(unmarkedFit-class), 60
- unmarkedFitGMM-class
(unmarkedFit-class), 60
- unmarkedFitList-class, 63

- unmarkedFitMPois-class
 - (unmarkedFit-class), 60
- unmarkedFitOccu-class
 - (unmarkedFit-class), 60
- unmarkedFitPCO-class
 - (unmarkedFit-class), 60
- unmarkedFitPCount-class
 - (unmarkedFit-class), 60
- unmarkedFrame, 4, 21, 41, 65, 67, 68, 70, 71, 73, 75, 77
- unmarkedFrame-class, 66, 70, 71, 73, 75, 77
- unmarkedFrame-class, 67
- unmarkedFrameDS, 17, 66, 69, 79
- unmarkedFrameDS-class
 - (unmarkedFrame-class), 67
- unmarkedFrameGDS, 21, 26
- unmarkedFrameGDS (unmarkedMultFrame), 78
- unmarkedFrameGDS-class
 - (unmarkedFrame-class), 67
- unmarkedFrameGMM, 29, 30
- unmarkedFrameGMM (unmarkedMultFrame), 78
- unmarkedFrameGMM-class
 - (unmarkedFrame-class), 67
- unmarkedFrameMPois, 38, 70
- unmarkedFrameMPois-class
 - (unmarkedFrame-class), 67
- unmarkedFrameOccu, 41, 42, 66, 72
- unmarkedFrameOccu-class
 - (unmarkedFrame-class), 67
- unmarkedFramePCO, 49–51, 74
- unmarkedFramePCO-class
 - (unmarkedFrame-class), 67
- unmarkedFramePCount, 47, 48, 66, 76
- unmarkedFramePCount-class
 - (unmarkedFrame-class), 67
- unmarkedModSel-class (modSel), 36
- unmarkedMultFrame, 9, 10, 22, 78
- unmarkedMultFrame-class
 - (unmarkedFrame-class), 67
- unmarkedRanef-class, 55, 56, 81
- update, unmarkedFit-method
 - (unmarkedFit-class), 60
- update, unmarkedFitColExt-method
 - (unmarkedFit-class), 60
- update, unmarkedFitGMM-method
 - (unmarkedFit-class), 60
- update, unmarkedFitPCO-method
 - (unmarkedFit-class), 60
- vcov, 39
- vcov, linCombOrBackTrans-method
 - (vcov-methods), 82
- vcov, unmarkedEstimate-method
 - (vcov-methods), 82
- vcov, unmarkedFit-method (vcov-methods), 82
- vcov-methods, 82
- woodthrush (birds), 8
- xyplot, 81
- yearlySiteCovs (unmarkedMultFrame), 78
- yearlySiteCovs, unmarkedMultFrame-method
 - (unmarkedMultFrame), 78
- yearlySiteCovs<- (unmarkedMultFrame), 78
- yearlySiteCovs<- , unmarkedMultFrame-method
 - (unmarkedMultFrame), 78