Package ‘unmarked’

February 20, 2015

Version 0.10-6
Date 2015-01-27
Type Package
Title Models for Data from Unmarked Animals
Author Ian Fiske, Richard Chandler, David Miller, Andy Royle, Marc Kery, Jeff Hostetler
Maintainer Richard Chandler <rbchan@uga.edu>
Depends R (>= 2.12.0), methods, reshape, lattice, Rcpp (>= 0.8.0)
Suggests raster
Imports graphics, stats, utils, plyr
Description 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' 'colext.R' 'distsamp.R'
  'multinomPois.R' 'occu.R' 'occuRN.R' 'pcount.R' 'gmultmix.R'
  'pcountOpen.R' 'gdistsamp.R' 'unmarkedFitList.R'
  'unmarkedLinComb.R' 'ranef.R' 'boot.R' 'occuFP.R' 'gpcount.R'
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
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-01-27 20:56:24
R topics documented:

unmarked-package .................................................. 3
backTransform-methods ................................................. 7
birds ................................................................. 8
coef-methods ....................................................... 9
colext ............................................................ 9
confint-methods ................................................... 12
crossbill .......................................................... 13
cruz ............................................................. 15
csvToUMF ........................................................ 17
detFuns .......................................................... 18
distsamp .......................................................... 19
fitList .......................................................... 22
fitted-methods ................................................... 24
formatDistData ................................................... 25
formatMult ....................................................... 26
formatWideLong .................................................. 27
frogs ........................................................... 28
gdistsamp ....................................................... 29
getB-methods ................................................... 32
getFP-methods .................................................. 32
getP-methods ................................................... 32
gf .............................................................. 33
gmultmix ........................................................ 33
gpcount ........................................................ 36
imputeMissing ................................................... 38
issj .............................................................. 39
lambda2psi ........................................................ 40
linearComb-methods .......................................... 41
linetran ........................................................ 42
mallard .......................................................... 43
masspcru ........................................................ 43
modSel .......................................................... 44
multinomPois .................................................. 46
nonparboot-methods ........................................ 48
occu ........................................................... 49
occuFP .......................................................... 51
occuRN .......................................................... 53
ovendata ......................................................... 55
parboot .......................................................... 56
pcount .......................................................... 57
pcountOpen ...................................................... 60
piFuns .......................................................... 64
pointtran ......................................................... 65
predict-methods ............................................... 66
ranef-methods .................................................. 66
SE-methods ...................................................... 68
unmarked-package

Models for Data from Unmarked Animals

Description

Fits hierarchical models of animal occurrence and abundance to data collected on species that may be detected imperfectly. Models 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: 0.10-5
License: GPL (>= 3)

Overview of Model-fitting Functions:

- `occu` fits occurrence models with no linkage between abundance and detection (MacKenzie et al.
occuRN fits abundance models to presence/absence data by exploiting the link between detection probability and abundance (Royle and Nichols 2003).

occuFP fits occupancy models to data characterized by false negatives and false positive detections (e.g., Royle and Link [2006] and Miller et al. [2011]).


pcount fits N-mixture models (aka binomial mixture models) to repeated count data (Royle 2004a, Kery 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.

gpcount fits the generalized N-mixture model described by Chandler et al. (2011) to repeated count data collected using the robust design.

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 (e.g., 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

\[ \tilde{x_1} + x_2 + \ldots + x_n \tilde{x_1} + x_2 + \ldots + x_n \]

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 colex, 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 K\'ery
References


Examples

```r
## 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, 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
sum(z) # Actual

# Model-averaged prediction and plots
# psi in each habitat type
newdata1 <- data.frame(habitat=c('A', 'B'), veght=0)
Epsil1 <- predict(fms, type="state", newdata=newdata1)
with(Epsil1, {
    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))
Eps12 <- 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, Eps12, type="l", lwd=2, ylim=c(0,1),
     xlab="Vegetation height (standardized)",
     ylab=expression(paste("Probability of occurrence (", psi, ")")))
lines(lower ~ veght, Eps12, col=gray(0.7))
lines(upper ~ veght, Eps12, 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)

Description
Methods for function backTransform in Package 'unmarked'

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 co-
  variates 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

```r
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

```r
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


Examples

```r
data(birds)
```
Methods for Function `coef` in Package `unmarked`

**Description**

Extract coefficients

**Usage**

```r
## 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 dynamic occupancy model of MacKenzie et. al (2003)*

**Description**

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

**Usage**

```r
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 (i.e., 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 `smoothedNmean` and `projectedNmean` slots (see examples).

Value

unmarkedFitColExt object describing model fit.

References


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=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=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=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(numPrimary) # summarize
fm <- colext(-1, -1, -1, -1, umf) # fit a model
fm

## Not run:
# Real data
data(frogs)
Methods for Function `confint` in Package 'unmarked'

Description

Methods for function `confint` in Package 'unmarked'

Usage

```r
## 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

---

crossbill  

**Description**

267 1-kmsq quadrats were surveyed 3 times per year during 1999-2007.

**Usage**

data(crossbill)

**Format**

A data frame with 267 observations on the following 58 variables.

- **id**: Plot ID
- **ele**: Elevation
- **forest**: Percent forest cover
- **surveys**: a numeric vector
- **det991**: Detection data for 1999, survey 1
- **det992**: Detection data for 1999, survey 2
- **det993**: Detection data for 1999, survey 3
- **det001**: Detection data for 2000, survey 1

**Detection/non-detection data on the European crossbill (Loxia curvirostra)**
det002 a numeric vector
det003 a numeric vector
det011 a numeric vector
det012 a numeric vector
det013 a numeric vector
det021 a numeric vector
det022 a numeric vector
det023 a numeric vector
det031 a numeric vector
det032 a numeric vector
det033 a numeric vector
det041 a numeric vector
det042 a numeric vector
det043 a numeric vector
det051 a numeric vector
det052 a numeric vector
det053 a numeric vector
det061 a numeric vector
det062 a numeric vector
det063 Detection data for 2006, survey 3
det071 Detection data for 2007, survey 1
det072 Detection data for 2007, survey 2
det073 Detection data for 2007, survey 3
date991 Day of the season for 1999, survey 1
date992 Day of the season for 1999, survey 2
date993 Day of the season for 1999, survey 3
date001 Day of the season for 2000, survey 1
date002 a numeric vector
date003 a numeric vector
date011 a numeric vector
date012 a numeric vector
date013 a numeric vector
date021 a numeric vector
date022 a numeric vector
date023 a numeric vector
date031 a numeric vector
date032 a numeric vector
cruz

date033  a numeric vector
date041  a numeric vector
date042  a numeric vector
date043  a numeric vector
date051  a numeric vector
date052  a numeric vector
date053  a numeric vector
date061  a numeric vector
date062  a numeric vector
date063  a numeric vector
date071  a numeric vector
date072  a numeric vector
date073  Day of the season for 2007, survey 3

Source

Schmid, H. N. Zbinden, and V. Keller. 2004. Überwachung der Bestandsentwicklung häufiger Brutvogel in der Schweiz, Swiss Ornithological Institute Sempach Switzerland

See Also

Switzerland for corresponding covariate data defined for all 1-kmsq pixels in Switzerland. Useful for making species distribution maps.

Examples

data(crossbill)
str(crossbill)

---

cruz  

*Landscape data for Santa Cruz Island*

---

Description

Spatially-referenced elevation, forest cover, and vegetation data for Santa Cruz Island.

Usage

data(cruz)
Format

A data frame with 2787 observations on the following 5 variables.

- **x**: Easting (meters)
- **y**: Northing (meters)
- **elevation**: a numeric vector, meters
- **forest**: a numeric vector, proportion cover
- **chaparral**: a numeric vector, proportion cover

Details

The resolution is 300x300 meters.

The Coordinate system is EPSG number 26911

**NAD_1983_UTM_Zone_11N Projection**: Transverse_Mercator False_Easting: 500000.000000 False_Northing: 0.000000 Central_Meridian: -117.000000 Scale_Factor: 0.999600 Latitude_Of_Origin: 0.000000 Linear Unit: Meter GCS_North_American_1983 Datum: D_North_American_1983

Source

Brian Cohen of the Nature Conservancy helped prepare the data

References


Examples

data(cruz)
str(cruz)

levelplot(elevation ~ x + y, cruz, aspect="iso",
  col.regions=terrain.colors(100))

if(require(raster)) {
elev <- rasterFromXYZ(cruz[,1:3],
  crs="+proj=utm +zone=11 +ellps=GRS80 +datum=NAD83 +units=m +no_defs")
elev
plot(elev)
}
**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**

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

**Arguments**

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

**Details**

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

The .csv file is formatted as follows:

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

**Value**

an unmarkedFrame object

**Author(s)**

Ian Fiske <ianfiske@gmail.com>
Examples

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

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

# convert them!

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

detFuns

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

Description

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

Usage

- `gxhn(x, sigma)`
- `gxexp(x, rate)`
- `gxhaz(x, shape, scale)`
- `dxhn(x, sigma)`
- `dxexp(x, rate)`
- `dxhaz(x, shape, scale)`
- `drhn(r, sigma)`
- `drexpr(r, rate)`
- `dhrhaz(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, engine=c("C", "R"), rel.tol=0.001, ...)

Arguments

formula Double right-hand formula describing detection covariates followed by abundance covariates. ~1 ~1 would be a null model.
data object of class unmarkedFrameDS, containing response matrix, covariates, distance interval cut points, survey type ("line" or "point"), transect lengths (for survey = "line"), and units ("m" or "km") for cut points and transect lengths. See example for set up.
keyfun One of the following detection functions: "halfnorm", "hazard", "exp", or "uniform." See details.
output Model either "density" or "abund"
unitsOut  Units of density. Either "ha" or "kmsq" for hectares and square kilometers, respectively.
starts  Vector of starting values for parameters.
method  Optimization method used by optim.
se  logical specifying whether or not to compute standard errors.
engine  Use code written in C++ or R
rel.tol  Requested relative accuracy of the integral, see integrate
...  Additional arguments to optim, 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 allows for modeling both abundance and detection function parameters.

The latent transect-level abundance distribution \( f(N|\theta) \) assumed to be Poisson with mean \( \lambda \) (but see gdistsamp for alternatives).

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

Parameters \( \lambda \) and \( \sigma \) 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 <rbchan@uga.edu>

References


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, {
    unmarkedFrameOS(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)
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

# Half-normal. 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 examples

# Analysis of the Island Scrub-jay data.
# See Sillett et al. (In press)

data(issj)
str(issj)

jayumf <- unmarkedFrameDS(y = as.matrix(issj[, 1:3]),
  siteCovs = data.frame(scale(issj[, c("elevation", "forest", "chaparral")]),
  dist.breaks = c(0, 100, 200, 300), unitsIn = "m", survey = "point")

(fm1jay <- distsamp(~chaparral ~chaparral, jayumf))

## 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.
(fm1 <- distsamp(~ 1 ~ 1, ptUMF))
hist(fm1, ylim=c(0, 0.07), xlab="Distance (m)"

# effective radius
sig <- exp(coef(fm1, type="det"))

## End(Not run)

---

**fitList**

*constructor of unmarkedFitList objects*

**Description**

Organize models for model selection or model-averaged prediction.

**Usage**

fitList(..., fits)
fitList

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 (e.g., 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 <rbchan@uga.edu>

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,
  length = 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 argument name "fits" must be included in call.
models <- list(Null=fm1, .area=fm2, area.=fm3)
fmList <- fitList(fits = models)

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

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

# Model selection
modSel(fmList, nullmod="Null")
Methods for Function fitted in Package 'unmarked'

Description

Extracted fitted values from a fitted model.

Usage

```r
## 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**

Bin distance data

**Description**

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

**Usage**

```r
formatDistData(distData, distCol, transectNameCol, dist.Nbreaks, 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.
- **occasionCol**: optional character. If transects were visited more than once, this can be used to format data for `gdistsamp`. 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**

Convert between wide and long data formats.

**Description**

Convert a data.frame between wide and long formats.

**Usage**

```r
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 \texttt{dfin} need to be in the correct order. \texttt{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.

\texttt{formatWide} requires particular names for the columns. The column order for \texttt{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

\begin{verbatim}
28 frogs 2001 Delaware North American Amphibian Monitoring Program Data
\end{verbatim}

Description

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

Usage

data(frogs)

Format

\texttt{pcru.y} matrix of observed calling indices for pcru
\texttt{pcru.bin} matrix of detections for pcru
\texttt{pcru.data} array of covariates measured at the observation-level for pcru
\texttt{pfer.y} matrix of observed calling indices for pfer
\texttt{pfer.bin} matrix of detections for pfer
\texttt{pfer.data} array of covariates measured at the observation-level for pfer
**gdistsamp**

**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 × observation), with the 3rd dimension corresponding to separate covariates.

**Source**

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

**References**


**Examples**

```r
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**

```r
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 $\phi$. 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 $i = 1, 2, ..., R$ transects on $t = 1, 2, ..., T$ 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 phi, 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 colext methods of formatting data upon which it is based.)

Author(s)
Richard Chandler <rbchan@uga.edu>

References
gdistsamp

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
ty <- 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))
    }
  }
}  # convert array to matrix

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", K=50)

summary(m1)

backTransform(m1, type="lambda")
backTransform(m1, type="phi")
getP-methods

Methods for Function \texttt{getP} in Package ‘unmarked’

Description

Methods for function \texttt{getP} in Package ‘unmarked’. These methods return a matrix of detection probabilities.

Methods

\begin{itemize}
\item \texttt{object = "unmarkedFit"} A fitted model object
\item \texttt{object = "unmarkedFitDS"} A fitted model object
\item \texttt{object = "unmarkedFitMPois"} A fitted model object
\item \texttt{object = "unmarkedFitGMM"} A fitted model object
\end{itemize}
**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**

**Examples**
data(gf)  
str(gf.data)  
str(gf.obs)

---

**gmultmix**  
*Generalized multinomial N-mixture model*

**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. \( \text{mixture} = \text{"P"} \) or \( \text{mixture} = \text{"NB"} \) select the Poisson or negative binomial distribution respectively. The mean of \( M_i \) is \( \lambda_i \). If \( M_i \sim NB \), then an additional parameter, \( \alpha \), describes dispersion (lower \( \alpha \) implies higher variance).

The number of individuals available for detection at time \( j \) is modeled as binomial: \( N_{ij} \sim \text{Binomial}(M_i, \phi_{ij}) \).

The detection process is modeled as multinomial: \( y_{it} \sim \text{Multinomial}(N_{it}, \pi_{ij}) \), where \( \pi_{ijt} \) 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 \( \lambda, \phi \) 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

In the case where availability for detection is due to random temporary emigration, population density at time \( j \), \( D(i,j) \), can be estimated by \( N(i,j)/\text{plotArea} \).

This model is also applicable to sampling designs in which the local population size is closed during the \( J \) repeated counts, and availability is related to factors such as the probability of vocalizing. In this case, density can be estimated by \( M(i)/\text{plotArea} \).

If availability is a function of both temporary emigration and other processes such as song rate, then density cannot be directly estimated, but inference about the super-population size, \( M(i) \), is possible.
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 `unmarkedFrameGPM`.

**Author(s)**

Richard Chandler <rbchan@uga.edu> and Andy Royle

**References**


**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**

```r
# 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))  # Local population size
M <- rpois(n, lam)  # Individuals available for detection
N <- matrix(NA, n, T)

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)

gpcount  Generalized binomial N-mixture model for repeated count data

Description

Fit the model of Chandler et al. (2011) to repeated count data collected using the robust design. This model allows for inference about population size, availability, and detection probability.

Usage

gpcount(lambdaformula, phiformula, pformula, data, 
mixture = c("P", "NB"), K, starts, method = "BFGS", se = TRUE, 
engine = c("C", "R"), ...)

Arguments

lambdaformula  Right-hand sided formula describing covariates of abundance.
phiformula     Right-hand sided formula describing availability covariates
pformula       Right-hand sided formula for detection probability covariates
data            An object of class unmarkedFrameGPC
mixture        Either "P" or "NB" for Poisson and negative binomial distributions
K              The maximum possible value of M, the super-population size.
starts         Starting values
method         Optimization method used by optim
se             Logical. Should standard errors be calculated?
engine         Either "C" or "R" for the C++ or R versions of the likelihood. The C++ code is faster, but harder to debug.
...            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 \texttt{mixture} argument. The expected value of \( M_i \) is \( \lambda_i \). If \( M_i \sim NB \), then an additional parameter, \( \alpha \), describes dispersion (lower \( \alpha \) implies higher variance).

The number of individuals available for detection at time \( j \) is modeled as binomial: \( N_{ij} \sim Binomial(M_i, \phi_{ij}) \).

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

Parameters \( \lambda, \phi \), and \( p \) can be modeled as linear functions of covariates using the log, logit and logit links respectively.

Value
An object of class unmarkedFitGPC

Note
In the case where availability for detection is due to random temporary emigration, population density at time \( j \), \( D(i,j) \), can be estimated by \( N(i,j)/\text{plotArea} \).

This model is also applicable to sampling designs in which the local population size is closed during the \( J \) repeated counts, and availability is related to factors such as the probability of vocalizing. In this case, density can be estimated by \( M(i)/\text{plotArea} \).

If availability is a function of both temporary emigration and other processes such as song rate, then density cannot be directly estimated, but inference about the super-population size, \( M(i) \), is possible.

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 \texttt{unmarkedFrameGPC}

Author(s)
Richard Chandler <rbchan@uga.edu>

References


See Also
\texttt{gmultmix, gdistsamp, unmarkedFrameGPC}
Examples

```r
set.seed(54)

nSites <- 20
nVisits <- 4
nReps <- 3

lambda <- 5
phi <- 0.7
p <- 0.5

M <- rpois(nSites, lambda) # super-population size

N <- matrix(NA, nSites, nVisits)
y <- array(NA, c(nSites, nReps, nVisits))
for(i in 1:nVisits) {
  N[,i] <- rbinom(nSites, M, phi) # population available during visit j
}
colMeans(N)

for(i in 1:nSites) {
  for(j in 1:nVisits) {
    y[i,,j] <- rbinom(nReps, N[i,j], p)
  }
}

ym <- matrix(y, nSites)
ym[1,] <- NA
ym[2, 1:nReps] <- NA
ym[3, (nReps+1):(nReps+nReps)] <- NA
umf <- unmarkedFrameGPC(y=ym, numPrimary=nVisits)

## Not run:
fmu <- gpcount(~1, ~1, ~1, umf, K=40, control=list(trace=TRUE, REPORT=1))
backTransform(fmu, type="lambda")
backTransform(fmu, type="phi")
backTransform(fmu, type="det")

## End(Not run)
```

---

**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

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

Arguments

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

Value

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

Author(s)

Ian Fiske

Examples

```r
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.11 <- imputeMissing(pcruUMF)
pcruUMF.12 <- imputeMissing(pcruUMF, whichCovs = 2)
```

---

**issj**

*Distance-sampling data for the Island Scrub Jay (Aphelocoma insularis)*

Description

Data were collected at 307 survey locations ("point transects") on Santa Cruz Island, California during the Fall of 2008. The distance data are binned into 3 distance intervals [0-100], (100-200], and (200-300]. The coordinates of the survey locations as well as 3 habitat covariates are also included.

Usage

```r
data(issj)
```
Format

A data frame with 307 observations on the following 8 variables.

- `issj[0-100]` Number of individuals detected within 100m
- `issj[100-200]` Detections in the interval (100-200m)
- `issj[200-300]` Detections in the interval (200-300m)
- `x` Easting (meters)
- `y` Northing (meters)
- `elevation` Elevation in meters
- `forest` Forest cover
- `chaparral` Chaparral cover

References


See Also

Island-wide covariates are also available `cruz`

Examples

```r
data(issj)
str(issj)
head(issj)

umf <- unmarkedFrameDS(y=as.matrix(issj[,1:3]), siteCovs=issj[,6:8],
  dist.breaks=c(0,100,200,300), unitsIn="m", survey="point")
summary(umf)
```

lambda2psi

<table>
<thead>
<tr>
<th>lambda2psi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert Poisson mean (lambda) to probability of occurrence (psi).</td>
</tr>
</tbody>
</table>

Description

Abundance and occurrence are fundamentally related.

Usage

`lambda2psi(lambda)`

Arguments

`lambda` Numeric vector with values $\geq 0$
**linearComb-methods**

**Value**

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

**See Also**

`pcount, multinomPois, distsamp`

**Examples**

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

---

**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**

```r
data(ovendata)
ovenFrame <- unmarkedFrameMMPois(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, 1, 0, 0, 1, 0, 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")
```
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()
luMF <- 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

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


Examples

data(masspcru)
str(masspcru)

modSel  

Model selection results from an unmarkedFitList

Description

Model selection results from an unmarkedFitList

Arguments

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

Value

A S4 object with the following slots

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

Names matrix referencing column names of estimates (row 1) and standard errors (row 2).

Note

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

Author(s)

Richard Chandler <rbchan@uga.edu>

References


Examples

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

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

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

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

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

coef(ms) # Estimates only
multinomPois

Description

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

Usage

```r
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 `optim`.
- `se`: logical specifying whether or not to compute standard errors.
- `...`: Additional arguments to `optim`, 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


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], cp, 1-sum(cp))[1:3]
}

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

# 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(fml, type="det")

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

# Empirical Bayes estimates of abundance at first 25 sites
# Very low uncertainty because p is very high
plot(ranef(fml, 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". For occu and occuRN both sites and occasions are re-sampled. For all other fitting functions, only sites are re-sampled.

**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 = "unmarkedFit0Occu") Obtain nonparametric bootstrap samples for a occu fits.
signature(object = "unmarkedFit0OccuRN") Obtain nonparametric bootstrap samples for a occuRN fits.
signature(object = "unmarkedFitPCount") Obtain nonparametric bootstrap samples for a pcount fits.
Examples

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

```r
## 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

```r
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 `unmarkedFrameOccu` 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 `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

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

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

The observation process is modeled as

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

Covariates of $\psi_i$ 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


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
occuFP <- data.frame(obsvar1 = rnorm(numSites(pferUMF) * obsNum(pferUMF)))

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

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

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

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

# Empirical Bayes estimates of proportion of sites occupied
re <- ranef(fm)
sum(bup(re, stat="mode"))

---

occuFP Fit occupancy models when false positive detections occur (e.g., Royle and Link [2006] and Miller et al. [2011])

Description

This function fits the single season occupancy model while allowing for false positive detections.

Usage

occuFP(detformula = ~ 1, FPformula = ~ 1, Bformula = ~ 1, stateformula = ~ 1, data, starts, method="BFGS", se = TRUE, engine = "R", ...) 

Arguments

detformula formula describing covariates of detection.
FPformula formula describing covariates of false positive detection probability.
Bformula formula describing covariates of probability detections are certain.
stateformula formula describing covariates of occupancy.
data An unmarkedFrameOccuFP object
starts Vector of parameter starting values.
method Optimization method used by optim.
se Logical specifying whether or not to compute standard errors.
engine Currently only choice is R.
... Additional arguments to optim, such as lower and upper bounds
Details

See `unmarkedFrame` and `unmarkedFrameOccuFP` for a description of how to supply data to the data argument.

`occuFP` fits an extension of the standard single-season occupancy model (MacKenzie et al. 2002), which allows false positive detections. The occupancy status of a site is the same way as with the `occu` function, where stateformula is used to specify factors that lead to differences in occupancy probabilities among sites.

The observation process differs in that both false negative and false positive errors are modeled for observations. The function allows data to be of 3 types. These types are specified using in `unmarkedFrameOccuFP` as type. Occasions are specified to belong to 1 of the 3 data types and all or a subset of the data types can be combined in the same model.

For type 1 data, the detection process is assumed to fit the assumptions of the standard MacKenzie model where false negative probabilities are estimated but false positive detections are assumed not to occur. If all of your data is of this type you should use codeoccu to analyze data. The detection parameter p, which is modeled using the detformula is the only observation parameter for these data.

For type 2 data, both false negative and false positive detection probabilities are estimated. If all data is of this type the likelihood follows Royle and Link (2006). Both p (the true positive detection probability) and fp (the false positive detection probability described by fpformula) are estimated for occasions when this data type occurs.

For type 3 data, observations are assumed to include both certain detections (false positives assumed not to occur) and uncertain detections that may include false positive detections. When only this data type occurs, the estimator is the same as the multiple detection state model described in Miller et al. (2011). Three observation parameters occur for this data type: p - true positive detection probability, fp - false positive detection probability, and b - the probability a true positive detection was designated as certain.

When both type 1 and type 2 data occur, the estimator is equivalent to the multiple detection method model described in Miller et al. (2011). The frog data example in the same paper uses an analysis where type 1 (dipnet surveys) and type 3 (call surveys) data were used.

Data in the y matrix of the unmarked frame should be all 0s and 1s for type 1 and type 2 data. For type 3 data, uncertain detections are given a value of 1 and certain detections a value of 2.

Value

`unmarkedFitOccuFP` object describing the model fit.

Author(s)

David Miller

References


See Also

unmarked, unmarkedFrameOccFP, modSel, parboot

Examples

```r
n = 100
o = 10
o1 = 5
y = matrix(0,n,o)
p = .7
r = .5
fp = 0.05
y[1:(n*.5),(o-o1+1):o] <- rbinom((n-o1*.5),1,p)
y[1:(n*.5),1:(o-o1)] <- rbinom((o-o1)*n*.5,1,r)
y[(n*.5+1):n,(o-o1+1):o] <- rbinom((n-o1*.5),1,fp)
type <- c((o-o1),o1,0)  ## vector with the number of each data type
site <- c(rep(1,n*.5*.8),rep(0,n*.5*.2),rep(1,n*.5*.2),rep(0,n*.8*.5))
occ <- matrix(c(rep(0,n*(o-o1)),rep(1,n-o1)),n,o)
site <- data.frame(habitat = site)
occ <- list(METH = occ)

umf1 <- unmarkedFrameOccFP(y,site,occ, type = type)

m1 <- occFP(detformula = ~ METH, FPformula = ~1,
           stateformula = ~ habitat, data = umf1)
predict(m1, type = 'fp')
coef(m1)
confint(m1, type = 'det')
```

**Description**

Fit the occupancy model of Royle and Nichols (2003)

**Usage**

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

- **formula**: double right-hand side formula describing covariates of detection and abundance, 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 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 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 \( \lambda_i \) 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

Examples

```r
## 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

```r
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

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 <rbchan@uga.edu>

See Also
ranef
Fit the N-mixture model of Royle (2004)

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)
  freemTukey <- sum((sqrt(observed) - sqrt(expected))^2)
  out <- c(SSE=sse, Chisq=chisq, freemTukey=freemTukey)
  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

\[ \text{pcount} \text{(formula, data, K, mixture=c("P", "NB", "ZIP"),}
\]

\[ \text{starts, method="BFGS", se=TRUE, engine=c("C", "R"), ...)} \]

Arguments

\begin{itemize}
\item \textbf{formula} \quad \text{Double right-hand side formula describing covariates of detection and abundance, in that order}
\item \textbf{data} \quad \text{an unmarkedFramePCount object supplying data to the model.}
\item \textbf{K} \quad \text{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.}
\item \textbf{mixture} \quad \text{character specifying mixture: "P", "NB", or "ZIP".}
\item \textbf{starts} \quad \text{vector of starting values}
\item \textbf{method} \quad \text{Optimization method used by \texttt{optim}.}
\item \textbf{se} \quad \text{logical specifying whether or not to compute standard errors.}
\item \textbf{engine} \quad \text{Either "C" or "R" to use fast C++ code or native R code during the optimization.}
\item \ldots \quad \text{Additional arguments to \texttt{optim}, such as lower and upper bounds}
\end{itemize}

Details

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

See \texttt{unmarkedFramePCount} for a description of how to format data for \texttt{pcount}.

This function fits the latent N-mixture model for point count data (Royle 2004, Kery 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 \texttt{mixture} argument, \texttt{mixture = "P"}, \texttt{mixture = "NB"}, or \texttt{mixture = "ZIP"} respectively. For the first two distributions, the mean of \( N_i \) is \( \lambda_i \). If \( N_i \sim NB \), then an additional parameter, \( \alpha \), describes dispersion (lower \( \alpha \) 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 \( \lambda_i \) use the log link and covariates of \( p_{ij} \) use the logit link.

Value

\texttt{unmarkedFit object describing the model fit.}

Author(s)

Ian Fiske and Richard Chandler
References


See Also

`unmarkedFramePCount`, `pcountOpen`, `ranef`, `parboot`

Examples

```r
# 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
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 models of Dail and Madsen and extensions**

---

**Description**

Fit the models of Dail and Madsen (2011) and Hostetler and Chandler (in press), which are generalized forms 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", "ricker", "gompertz"), fix=c("none", "gamma", "omega"), 
starts, method = "BFGS", se = TRUE, immigration = FALSE, 
iotaformula =~ 1, ...)
```

**Arguments**

- `lambdaformula`: Right-hand sided formula for initial abundance
- `gammaformula`: Right-hand sided formula for recruitment rate (when dynamics is "constant", "autoreg", or "notrend") or population growth rate (when dynamics is "trend", "ricker", or "gompertz")
- `omegaformula`: Right-hand sided formula for apparent survival probability (when dynamics is "constant", "autoreg", or "notrend") or equilibrium abundance (when dynamics is "ricker" or "gompertz")
- `pformula`: Right-hand sided formula for detection probability
- `data`: An object of class `unmarkedFramePCO`. 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). "ricker" and "gompertz" are models for density-dependent population growth. "ricker" is the Ricker-logistic model, N[i,t] = N[i,t-1]*exp(gamma*(1-N[i,t-1]/omega)), where gamma is the maximum instantaneous population growth rate (normally referred to as r) and omega is the equilibrium abundance (normally referred to as K). "gompertz" is a modified version of the Gompertz-logistic model, N[i,t] = N[i,t-1]*exp(gamma*(1-log(N[i,t-1]+1)/log(omega+1))), where the interpretations of gamma and omega are similar to in the Ricker model.

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.

immigration logical specifying whether or not to include an immigration term (iota) in population dynamics.

iotaformula Right-hand sided formula for average number of immigrants to a site per time step

... additional arguments to be passed to optim.

Details

These models generalize the Royle (2004) N-mixture model by relaxing the closure assumption. The models include two or three additional parameters: gamma, either the recruitment rate (births and immigrations), the finite rate of increase, or the maximum instantaneous rate of increase; omega, either the apparent survival rate (deaths and emigrations) or the equilibrium abundance (carrying capacity); and iota, the number of immigrants per site and year. 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".

When immigration is set to FALSE (the default), iota is not modeled. When immigration is set to TRUE and dynamics is set to "autoreg", the model will separately estimate birth rate (gamma) and number of immigrants (iota). When immigration is set to TRUE and dynamics is set to "trend", "ricker", or "gompertz", the model will separately estimate local contributions to population growth (gamma and omega) and number of immigrants (iota).

The latent abundance distribution, f(N|θ) 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 Ni is λi. If Ni ~ NB, then an additional parameter, α, describes dispersion (lower α implies higher variance). For the ZIP distribution, the mean is λi(1 - ψ), where psi is the zero-inflation parameter.

For "constant", "autoreg", or "notrend" dynamics, the latent abundance state following the initial sampling period arises from a Markovian process in which survivors are modeled as S_{it} ~
Binomial($N_{it-1}$, $\omega_{it}$), and recruits follow $G_{it} \sim \text{Poisson}(\gamma_{it})$. Alternative population dynamics can be specified using the dynamics and immigration arguments.

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

$\lambda_{i}$, $\gamma_{it}$, and $\iota_{it}$ are modeled using the log link. $p_{ijt}$ is modeled using the logit link. $\omega_{it}$ is either modeled using the logit link (for "constant", "autoreg", or "notrend" dynamics) or the log link (for "ricker" or "gompertz" dynamics). For "trend" dynamics, $\omega_{it}$ is not modeled.

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 <rbchan@uga.edu> and Jeff Hostetler

References


See Also

pcount, unmarkedFramePCO

Examples

```r
## 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"))
on <- plogis(coef(m1, type="omega"))
p <- plogis(coef(m1, type="det"))

## Not run:
# 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)))

# 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))
```
## piFuns

### Compute multinomial cell probabilities

#### Description

Compute the cell probabilities used in the multinomial-Poisson models `multinomPois` and `gmultmix`.

#### 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

```r
(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)
  # Code implementation
}````
```r
J <- ncol(p)
pj <- matrix(NA, M, J)
p[1,] <- p[,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) {
p[i,] <- pi[i, i - 1]/p[i, 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

```r
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

```r
data(pointtran)
pointtran

# Format for distsamp()
ptUMF <- with(pointtran, {
  unmarkedFrameDS(y = cbind(dc1, dc2, dc3, dc4, dc5),
  siteCovs = data.frame(area, habitat),
  dist.b 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 'psi', 'col', 'ext', or 'det'.
- `signature(object = "unmarkedFitGMM")` "type" must be 'lambda', 'psi', 'det'
- `signature(object = "unmarkedFitList")` "type" depends upon the fitted models

### ranef-methods

**Methods for Function ranef in Package unmarked**

**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, \( Pr(z_i = 1 | y_{ij}, \hat{\psi}_i, \hat{\rho}_{ij}) \)
- `signature(object = "unmarkedFitOccuRN")` Computes the conditional abundance distribution given the data and the estimates of the fixed effects, \( Pr(N_i = k | y_{ij}, \hat{\lambda}_i, \hat{\rho}_{ij})k = 0, 1, \ldots, K \)
- `signature(object = "unmarkedFitPCount")` \( Pr(N_i = k | y_{ij}, \hat{\lambda}_i, \hat{\rho}_{ij})k = 0, 1, \ldots, K \)
- `signature(object = "unmarkedFitMPois")` \( Pr(N_i = k | y_{ij}, \hat{\lambda}_i, \hat{\rho}_{ij})k = 0, 1, \ldots, K \)
signature(object = "unmarkedFitDS") $Pr(N_i = k|y_{i1:J}, \hat{\lambda}_i, \hat{\sigma}_i)k = 0, 1, \ldots, K$

signature(object = "unmarkedFitGMM") $Pr(M_i = k|y_{i1:J}, \hat{\lambda}_i, \hat{\phi}_{it}, \hat{\rho}_{ij,t})k = 0, 1, \ldots, K$

signature(object = "unmarkedFitGDS") $Pr(M_i = k|y_{i1:J}, \hat{\lambda}_i, \hat{\phi}_{it}, \hat{\sigma}_{it})k = 0, 1, \ldots, K$

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

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

**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). Eventually, we hope to add methods to account for the uncertainty of the hyperparameters.

Note also that the posterior mode appears to exhibit some bias as an estimator or abundance. Consider using the posterior mean instead, even though it will not be an integer in general. More simulation studies are needed to evaluate the performance of empirical Bayes methods for these models.

**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 consitutes 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 <rbchan@uga.edu>

**References**


**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))

# 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

```r
SSE(fit)
```

### Arguments
- **fit**: An unmarked fit object.

### Value
A numeric value for the models SSE.

### See Also
- `parboot`
Switzerland

Swiss landscape data

Description
Spatially-referenced data on elevation, forest cover, and water at a 1km-sq resolution.

Usage
data(Switzerland)

Format
A data frame with 42275 observations on the following 5 variables.
x Easting (m)
y Northing (m)
elevation a numeric vector (m)
forest a numeric vector (percent cover)
water a numeric vector (percent cover)

Details
Forest and water coverage (in percent area) was computed using the 1992-97 landcover dataset of the Swiss Federal Statistical Office (http://www.bfs.admin.ch). Median elevation (in metres) was computed using a median aggregation of the digital elevation model of the Swiss Federal Statistical Office.
x and y are the coordinates of the center of each 1km2 pixel.
The coordinate reference system is CH1903/LV03 (EPSG number: 21781).
These data can only be used for non-profit projects. Otherwise, written permission must be obtained from the Swiss Federal Statistical Office

Source
Swiss Federal Statistical Office (http://www.bfs.admin.ch)

Examples
data(Switzerland)
str(Switzerland)

levelplot(elevation ~ x + y, Switzerland, aspect="iso",
         col.regions=terrain.colors(100))

# Not run:
library(raster)
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
- getFP signature(object = "unmarkedFit"): calculates and extracts expected false positive detection probabilities
- getB signature(object = "unmarkedFit"): calculates and extracts expected probabilities a true positive detection was classified as certain
- 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**

```r
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
coe(fml, 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
unmarkedFrame

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)

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

coef(fl)
SE(fl)

ms <- modSel(fl, 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.frame` 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 covariates is ordered by site first, then observation number. Missing values are coded with `NA` in any of y, siteCovs, or obsCovs.

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

Several child classes of unmarkedFrame require 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

```r
# 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)

## Not run:
# Set up data for colext()
frogUMF <- formatMult(masspcru)
summary(frogUMF)

## End(Not run)

---

**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
unmarkedFrameDS

Examples

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

# Visualize it
plot(mallardUMF)
mallardUMF

# Summarize it
summary(mallardUMF)
str(mallardUMF)
numSites(mallardUMF)
numY(mallardUMF)
obsNum(mallardUMF)

# Extract components of data
gGetY(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  Organize data for the distance sampling model of Royle et al. (2004) fit by distsamp

Description

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

Usage

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

Arguments

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


See Also

unmarkedFrame-class, unmarkedFrame, distsamp
unmarkedFrameMPois

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
```

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 `data.frame` 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 `data.frames` or a `data.frame` 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

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=2, ncol=2, byrow=TRUE),
  x4 = matrix(c(
    'a', 'b',
    'a', 'b',
    'a', 'b',
    'a', 'b'),
    nrow=2, ncol=2, 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)
p1 <- matrix(NA, M, 3)
p1[, 1] <- p[, 1] * (1 - p[, 2])
p1[, 2] <- p[, 2] * (1 - p[, 1])
p1[, 3] <- p[, 1] * p[, 2]
  return(p1)
}

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

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

**unmarkedFrameOccu**

Organize data for the single season occupancy models fit by occu and occuRN

**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

y  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.

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 RxJ rows in site-major order.

mapInfo  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',
'a','b','c',
'c','a','b',
'b','c','a'), nrow=R, ncol=J, byrow=TRUE))

unmarkedFrameOccuFP

Organize data for the single season occupancy models fit by occuFP

Description

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

Usage

unmarkedFrameOccuFP(y, siteCovs=NULL, obsCovs=NULL, type, mapInfo)

Arguments

y 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.

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 RxJ rows in site-major order.

type A vector with 3 values designating the number of occasions where data is of type 1, type 2, and type 3 - see occuFP for more details about data types.

mapInfo Currently ignored

Details

unmarkedFrameOccuFP is the S4 class that holds data to be passed to the occu and occuRN model-fitting function.

Value

an object of class unmarkedFrameOccuFP
Create an object of class unmarkedFramePCO that contains data used by pcountOpen.

**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**

```r
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), 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, 1, 3, 2, 1, 1,
  1, 1, 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)
unmarkedFramePCount

Organize data for the N-mixture model fit by pcount

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
- **y**: 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.
- **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 RxJ rows in site-major order.
- **mapInfo**: 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

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

Create an unmarkedMultFrame, unmarkedFrameGMM, unmarkedFrameGDS, or unmarkedFrameGPC object

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, length)
unmarkedFrameGPC(y, siteCovs, obsCovs, numPrimary, yearlySiteCovs)

Arguments

y  A matrix of the observed data.
siteCovs  Data frame of covariates that vary at the site level.
**unmarkedMultFrame**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obsCovs</td>
<td>Data frame of covariates that vary within site-year-observation level.</td>
</tr>
<tr>
<td>numPrimary</td>
<td>Number of primary time periods (seasons in the multiseason model).</td>
</tr>
<tr>
<td>yearlySiteCovs</td>
<td>Data frame containing covariates at the site-year level.</td>
</tr>
<tr>
<td>type</td>
<td>Either &quot;removal&quot; or &quot;double&quot; for constant-interval removal sampling or double observer sampling. This should not be specified for other types of survey designs.</td>
</tr>
<tr>
<td>obsToY</td>
<td>A matrix specifying relationship between observation-level covariates and response matrix.</td>
</tr>
<tr>
<td>piFun</td>
<td>A function converting an MxJ matrix of detection probabilities into an MxJ matrix of multinomial cell probabilities.</td>
</tr>
<tr>
<td>dist.breaks</td>
<td>see unmarkedFrameGDS</td>
</tr>
<tr>
<td>survey</td>
<td>see unmarkedFrameGDS</td>
</tr>
<tr>
<td>unitsIn</td>
<td>see unmarkedFrameGDS</td>
</tr>
<tr>
<td>tlength</td>
<td>see unmarkedFrameGDS</td>
</tr>
</tbody>
</table>

**Details**

unmarkedMultFrame objects are used by `colext`.  
unmarkedFrameGMM objects are used by `gmultmix`.  
unmarkedFrameGDS objects are used by `gdistsamp`.  
unmarkedFrameGPC objects are used by `gpcount`.  

For a study with $M$ sites, $T$ years, and a maximum of $J$ observations per site-year, the data can be supplied in a variety of ways but are stored as follows. $y$ is an $M \times T \times J$ 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 yearlySite-Covs may be a misnomer is some cases.

See Also
`formatMult`, `colext`, `gmultmix`, `gpcount`

Examples

```r
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))
ocasions <- 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=ocasions),
  yearlySiteCovs=list(year=years),
  numPrimary=T)

umfGMM <- unmarkedFrameGMM(y=y,
  siteCovs = data.frame(site=site),
  obsCovs=list(occasion=ocasions),
  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 unmarkedFrameMPos
o2y <- diag(ncol(y))
```
unmarkedRanef-class

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 (lambda or psi). Simulation studies indicate that the posterior mode can exhibit (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


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 'unmarkedMultFrameGMM,numeric,missing,missing'

`x[i, j]`

## S4 method for signature 'unmarkedMultFrameGDS,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.
Methods

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, 72
  unmarkedEstimateList-class, 73
  unmarkedFit-class, 73
  unmarkedFitList-class, 76
  unmarkedFrame-class, 79
  unmarkedRanef-class, 95

*Topic datasets
  birds, 8
  crossbill, 13
  cruz, 15
  frogs, 28
  gf, 33
  issj, 39
  linetran, 42
  mallard, 43
  masspcru, 43
  ovendata, 55
  pointtran, 65
  Switzerland, 71

*Topic methods
  [-methods, 96
  backTransform-methods, 7
  coef-methods, 9
  confint-methods, 12
  fitted-methods, 24
  getB-methods, 32
  getFP-methods, 32
  getP-methods, 32
  linearComb-methods, 41
  nonparboot-methods, 48
  predict-methods, 66
  ranef-methods, 66
  SE-methods, 68
  simulate-methods, 69
  vcov-methods, 96

*Topic models
  coext, 9
  distsamp, 19
  gdistamp, 29
  multinomPois, 46
  occu, 49
  occuFP, 51
  occuRN, 53
  pcount, 57
  pcountOpen, 60

*Topic model
  gmultmix, 33

*Topic package
  unmarked-package, 3

*Topic utilities
  csvToUMF, 17
  imputeMissing, 38
  [,unmarkedEstimateList,ANY,ANY,ANY-method
  ([-methods), 96
  [,unmarkedFit,ANY,ANY,ANY-method
  ([-methods), 96
  [,unmarkedFrame,list,missing,missing-method
  ([-methods), 96
  [,unmarkedFrame,missing,numERIC,missing-method
  ([-methods), 96
  [,unmarkedFrame,numERIC,missing,missing-method
  ([-methods), 96
  [,unmarkedFrame,numERIC,numERIC,missing-method
  ([-methods), 96
  [,unmarkedFrameGDS,numERIC,missing,missing-method
  ([-methods), 96
  [,unmarkedFrameGMM,numERIC,missing,missing-method
  ([-methods), 96
  [,unmarkedFrameGPC,missing,numERIC,missing-method
  ([-methods), 96
  [,unmarkedFrameGPC,numERIC,missing,missing-method
  ([-methods), 96
  [,unmarkedFramePCO,missing,numERIC,missing-method
  ([-methods), 96
  [,unmarkedFramePCO,numERIC,missing,missing-method
  ([-methods), 96
  [,unmarkedMultFrame,missing,numERIC,missing-method
  ([-methods), 96

99
nonparboot, unmarkedFitCoExt-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitDS-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitGDS-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitGMM-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitMPois-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitMPCO-method
  (nonparboot-methods), 48
nonparboot, unmarkedFitPCount-method
  (nonparboot-methods), 48
nonparboot-methods, 48
numSites (unmarkedFrame-class), 79
numSites, unmarkedFrame-method
  (unmarkedFrame-class), 79
numY (unmarkedFrame-class), 79
numY, unmarkedFrame-method
  (unmarkedFrame-class), 79
obsCovs (unmarkedFrame-class), 79
obsCovs, unmarkedFrame-method
  (unmarkedFrame-class), 79
obsCovs <- (unmarkedFrame-class), 79
obsCovs <-, unmarkedFrame-method
  (unmarkedFrame-class), 79
obsNum (unmarkedFrame-class), 79
obsNum, unmarkedFrame-method
  (unmarkedFrame-class), 79
obsToY (unmarkedFrame-class), 79
obsToY <-, unmarkedFrame-method
  (unmarkedFrame-class), 79
occu, 3, 4, 24, 48, 49, 52, 70, 85–87
occuFP, 4, 51, 87, 88
occuRN, 4, 24, 48, 53, 70, 85–87
optim, 10, 20, 30, 34, 36, 46, 49, 51, 54, 58, 61, 73
ovendata, 55
parboot, 20, 50, 53, 56, 59, 70
parboot, unmarkedFit-method
  (unmarkedFit-class), 73
pcount, 4, 24, 41, 57, 62, 70, 91
pcountOpen, 4, 59, 60, 88, 89
pcru.bin (frogs), 28
pcru.data (frogs), 28
pcru.y (frogs), 28
pfer.bin (frogs), 28
pfer.data (frogs), 28
pfer.y (frogs), 28
piFuns, 35, 47, 64, 84, 93
plot, parboot, missing-method (parboot), 56
plot, profile, missing-method
  (unmarkedFit-class), 73
plot, unmarkedFit, missing-method
  (unmarkedFit-class), 73
plot, unmarkedFrame, missing-method
  (unmarkedFrame-class), 79
plot, unmarkedRanef, missing-method
  (unmarkedRanef-class), 95
pointtran, 65
powerAnalysis (unmarkedFrame-class), 79
powerAnalysis, formula, unmarkedFramePCount, numeric-method
  (unmarkedFrame-class), 79
predict, 7
predict, ANY-method (predict-methods), 66
predict, unmarkedFit-method
  (predict-methods), 66
predict, unmarkedFitCoExt-method
  (predict-methods), 66
predict, unmarkedFitGDS-method
  (predict-methods), 66
predict, unmarkedFitGMM-method
  (predict-methods), 66
predict, unmarkedFitList-method
  (predict-methods), 66
predict, unmarkedFitMPois-method
  (predict-methods), 66
predict, unmarkedFitPCO-method
  (predict-methods), 66
predict, unmarkedFitPCount-method
  (predict-methods), 66
predict-methods, 66
profile, unmarkedFit-method
  (unmarkedFit-class), 73
projected (unmarkedFit-class), 73
projected, unmarkedFitCoExt-method
projection (unmarkedFrame-class), 79
projection, unmarkedFrame-method
(unmarkedFrame-class), 79
ranef, 20, 56, 59, 95, 96
ranef (ranef-methods), 66
ranef, unmarkedFitColExt-method
(ranef-methods), 66
ranef, unmarkedFitDS-method
(ranef-methods), 66
ranef, unmarkedFitGDS-method
(ranef-methods), 66
ranef, unmarkedFitGMM-method
(ranef-methods), 66
ranef, unmarkedFitGMMorGDS-method
(ranef-methods), 66
ranef, unmarkedFitGPC-method
(ranef-methods), 66
ranef, unmarkedFitM-Pois-method
(ranef-methods), 66
ranef, unmarkedFitOccu-method
(ranef-methods), 66
ranef, unmarkedFitOccuFP-method
(ranef-methods), 66
ranef, unmarkedFitOccuRN-method
(ranef-methods), 66
ranef, unmarkedFitPCO-method
(ranef-methods), 66
ranef, unmarkedFitP-Count-method
(ranef-methods), 66
ranef-methods, 66
removalPIFun, 34, 46
removalPIFun (piFuns), 64
residuals, unmarkedFit-method
(unmarkedFit-class), 73
residuals, unmarkedFitOccu-method
(unmarkedFit-class), 73
residuals, unmarkedFitOccuFP-method
(unmarkedFit-class), 73
residuals, unmarkedFitOccuRN-method
(unmarkedFit-class), 73
sampleSize (unmarkedfit-class), 73
sampleSize, unmarkedFit-method
(unmarkedfit-class), 73
SE (SE-methods), 68
SE, linCombOrBackTrans-method
(SE-methods), 68
SE, unmarkedEstimate-method
(SE-methods), 68
SE, unmarkedFit-method
(SE-methods), 68
SE, unmarkedFitList-method
(unmarkedFitList-class), 76
SE, unmarkedModSel-method
(modSel), 44
SE-methods, 68
show, parboot-method
(parboot), 56
show, unmarkedBackTrans-method
(backTransform-methods), 7
show, unmarkedEstimate-method
(unmarkedEstimate-class), 72
show, unmarkedEstimateList-method
(unmarkedEstimateList-class), 73
show, unmarkedFit-method
(unmarkedFit-class), 73
show, unmarkedFrame-method
(unmarkedFrame-class), 79
show, unmarkedLinComb-method
(linearComb-methods), 41
show, unmarkedModSel-method
(modSel), 44
show, unmarkedMultiFrame-method
(unmarkedFrame-class), 79
show, unmarkedRanef-method
(unmarkedRanef-class), 95
sight2perpdist, 20, 69
simulate, unmarkedFitColExt-method
(simulate-methods), 69
simulate, unmarkedFitDS-method
(simulate-methods), 69
simulate, unmarkedFitGDS-method
(simulate-methods), 69
simulate, unmarkedFitGMM-method
(simulate-methods), 69
simulate, unmarkedFitGPC-method
(simulate-methods), 69
simulate, unmarkedFitM-Pois-method
(simulate-methods), 69
simulate, unmarkedFitOccu-method
(simulate-methods), 69
simulate, unmarkedFitOccuFP-method
(simulate-methods), 69
simulate, unmarkedFitOccuRN-method
(simulate-methods), 69
simulate, unmarkedFitPCO-method
(simulate-methods), 69
simulate, unmarkedFitP-Count-method
simulateMmethods, 69
siteCovs (unmarkedFrame-class), 79
siteCovs, unmarkedFrame-method (unmarkedFrame-class), 79
siteCovs<- (unmarkedFrame-class), 79
siteCovs<-, unmarkedFrame-method (unmarkedFrame-class), 79
smoothed (unmarkedFit-class), 73
smoothed, unmarkedFitCoExt-method (unmarkedFit-class), 73
SSE, 70
summary, unmarkedEstimate-method (unmarkedEstimate-class), 72
summary, unmarkedEstimateList-method (unmarkedEstimateList-class), 73
summary, unmarkedFit-method (unmarkedFit-class), 73
summary, unmarkedFitDS-method (unmarkedFit-class), 73
summary, unmarkedFitList-method (unmarkedFitList-class), 76
summary, unmarkedFrame-method (unmarkedFrame-class), 79
summary, unmarkedFrameDS-method (unmarkedFrame-class), 79
summary, unmarkedModSel-method (modSel), 44
summary, unmarkedMultFrame-method (unmarkedFrame-class), 79
Switzerland, 15, 71
unmarked, 17, 50, 53
unmarked (unmarked-package), 3
unmarkedEstimate (unmarkedEstimate-class), 72
unmarkedEstimate-class, 72
unmarkedEstimateList-class, 73
unmarkedFit, 73, 77, 80
unmarkedFit (unmarkedFit-class), 73
unmarkedFit-class, 73
unmarkedFitDS-class (unmarkedFit-class), 73
unmarkedFitGMM-class (unmarkedFit-class), 73
unmarkedFitList-class, 76
unmarkedFitMPois-class (unmarkedFit-class), 73
unmarkedFitOccu-class (unmarkedFit-class), 73
unmarkedFitOccuFP-class (unmarkedFit-class), 73
unmarkedFitPCO-class (unmarkedFit-class), 73
unmarkedFitPCount-class (unmarkedFit-class), 73
unmarkedFrame, 4, 25, 50, 52, 77, 79, 80, 82, 84, 86, 88, 89, 91
unmarkedFrame-class, 79
unmarkedFrameDS, 20, 78, 81, 93
unmarkedFrameDS-class (unmarkedFrame-class), 79
unmarkedFrameGDS, 25, 30
unmarkedFrameGDS (unmarkedMultFrame), 92
unmarkedFrameGDS-class (unmarkedFrame-class), 79
unmarkedFrameGMM, 34, 35
unmarkedFrameGMM (unmarkedMultFrame), 92
unmarkedFrameGMM-class (unmarkedFrame-class), 79
unmarkedFrameGPC, 35, 37
unmarkedFrameGPC (unmarkedMultFrame), 92
unmarkedFrameGPC-class (unmarkedFrame-class), 79
unmarkedFrameMPois, 47, 83
unmarkedFrameMPois-class (unmarkedFrame-class), 79
unmarkedFrameOccu, 49, 50, 54, 78, 85
unmarkedFrameOccu-class (unmarkedFrame-class), 79
unmarkedFrameOccuFP, 51–53, 87
unmarkedFramePCO, 60, 62, 88
unmarkedFramePCO-class (unmarkedFrame-class), 79
unmarkedFramePCount, 58, 59, 78, 91
unmarkedFramePCount-class (unmarkedFrame-class), 79
unmarkedModSel-class (modSel), 44
unmarkedMultFrame, 10, 27, 92
unmarkedMultFrame-class (unmarkedFrame-class), 79
unmarkedRanef-class, 66, 67, 95
update, unmarkedFit-method (unmarkedFit-class), 73
update, unmarkedFitColExt-method
   (unmarkedFit-class), 73
update, unmarkedFitGMM-method
   (unmarkedFit-class), 73
update, unmarkedFitPCO-method
   (unmarkedFit-class), 73
vcov, 48
vcov, linCombOrBackTrans-method
   (vcov-methods), 96
vcov, unmarkedEstimate-method
   (vcov-methods), 96
vcov, unmarkedFit-method (vcov-methods),
   96
vcov-methods, 96

woodthrush (birds), 8

xyplot, 95

yearlySiteCovs (unmarkedMultFrame), 92
yearlySiteCovs, unmarkedMultFrame-method
   (unmarkedMultFrame), 92
yearlySiteCovs<-(unmarkedMultFrame), 92
yearlySiteCovs<-, unmarkedMultFrame-method
   (unmarkedMultFrame), 92