Package 'spAbundance'

October 20, 2023

Type Package

Title Univariate and Multivariate Spatial Modeling of Species Abundance

Version 0.1.0

Description Fits single-species (univariate) and multi-species (multivariate) non-spatial and spatial abundance models in a Bayesian framework using Markov Chain Monte Carlo (MCMC). Spatial models are fit using Nearest Neighbor Gaussian Processes (NNGPs). Details on NNGP models are given in Datta, Banerjee, Finley, and Gelfand (2016) <doi:10.1080/01621459.2015.1044091> and Finley, Datta, and Banerjee (2020) <arXiv:2001.09111>. Fits single-species and multi-species spatial and non-spatial versions of generalized linear mixed models (Gaussian, Poisson, Negative Binomial), N-mixture models (Royle 2004 <doi:10.1111/j.0006-341X.2004.00142.x>) and hierarchical distance sampling models (Royle, Dawson, Bates (2004) <doi:10.1890/03-3127>). Multi-species spatial models are fit using a spatial factor modeling approach with NNGPs for computational efficiency.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://www.jeffdoser.com/files/spabundance-web

BugReports https://github.com/doserjef/spAbundance/issues

Depends R (>= 3.5.0)

Imports stats, coda, abind, RANN, lme4, foreach, doParallel, methods

Suggests testthat

NeedsCompilation yes

Author Jeffrey Doser [aut, cre],
Andrew Finley [aut]

Maintainer Jeffrey Doser <doserjef@msu.edu>

Repository CRAN

Date/Publication 2023-10-20 10:20:02 UTC

R topics documented:

abund . . . 3
bbsData . . . 8
bbsPredData . . . 9
dataNMixSim . . . 10
DS . . . 11
fitted.abund . . . 16
fitted.DS . . . 17
fitted.lfMsAbund . . . 18
fitted.lfMsDS . . . 18
fitted.lfMsNMix . . . 19
fitted.msAbund . . . 20
fitted.msDS . . . 21
fitted.msNMix . . . 21
fitted.NMix . . . 22
fitted.sfMsAbund . . . 23
fitted.sfMsDS . . . 24
fitted.sfMsNMix . . . 25
fitted.spAbund . . . 26
fitted.spDS . . . 26
fitted.spNMix . . . 27
fitted.svcAbund . . . 28
fitted.svcMsAbund . . . 29
hbefCount2015 . . . 29
lfMsAbund . . . 30
lfMsDS . . . 35
lfMsNMix . . . 42
msAbund . . . 47
msDS . . . 52
msNMix . . . 58
neonDWP . . . 63
neonPredData . . . 64
NMix . . . 65
ppcAbund . . . 69
predict.abund . . . 72
predict.DS . . . 75
predict.lfMsAbund . . . 78
predict.lfMsDS . . . 81
predict.lfMsNMix . . . 85
predict.msAbund . . . 89
predict.msDS . . . 91
predict.msNMix . . . 95
predict.NMix . . . 98
predict.sfMsAbund . . . 101
predict.sfMsDS . . . 104
predict.sfMsNMix . . . 109
predict.spAbund . . . 112

predict.spDS . . . 116
predict.spNMix . . . 120
predict.svcAbund . . . 123
predict.svcMsAbund . . . 127
sfMsAbund . . . 130
sfMsDS . . . 136
sfMsNMix . . . 144
simAbund . . . 150
simDS . . . 153
simMsAbund . . . 156
simMsDS . . . 159
simMsNMix . . . 162
simNMix . . . 166
spAbund . . . 169
spDS . . . 174
spNMix . . . 180
summary.abund . . . 186
summary.DS . . . 187
summary.lfMsAbund . . . 188
summary.lfMsDS . . . 189
summary.lfMsNMix . . . 190
summary.msAbund . . . 191
summary.msDS . . . 192
summary.msNMix . . . 193
summary.NMix . . . 194
summary.sfMsAbund . . . 195
summary.sfMsDS . . . 196
summary.sfMsNMix . . . 197
summary.spAbund . . . 198
summary.spDS . . . 199
summary.spNMix . . . 200
summary.svcAbund . . . 201
summary.svcMsAbund . . . 202
svcAbund . . . 203
svcMsAbund . . . 208
waicAbund . . . 214

Index **217**

---

abund *Function for Fitting Univariate Abundance GLMMs*

Description

Function for fitting univariate abundance generalized linear (mixed) models

Usage

```
abund(formula, data, inits, priors, tuning,
      n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
      n.omp.threads = 1, verbose = TRUE,
      n.report = 100, n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
      n.chains = 1, save.fitted = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, z, and offset. y is a vector, matrix, or data frame of the observed count values. If a vector, the values represent the observed counts at each site. If multiple replicate observations are obtained at the sites (e.g., sub-samples, repeated sampling over multiple seasons), y can be specified as a matrix or data frame with first dimension equal to the number of sites (J) and second dimension equal to the maximum number of replicates at a given site. covs is a list or data frame containing the variables used in the model. Each list element is a different covariate, which can be site-level or observation-level. Site-level covariates are specified as a vector of length J (or column in a data frame), while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicate observations at a given site. For zero-inflated Gaussian models, the tag z is used to specify the binary component of the zero-inflated model and should have the same length as y. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value, a vector with an offset for each site (e.g., if survey area differed in size), or a site x replicate matrix if more than one count is available at a given site.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta, kappa, sigma.sq.mu, and tau.sq. The value portion of each tag is the parameter's initial value. sigma.sq.mu is only relevant when including random effects in the model. kappa is only relevant when family = 'NB'. tau.sq is only relevant when family = 'Gaussian' or family = 'zi-Gaussian'. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.normal, kappa.unif, sigma.sq.mu.ig, and tau.sq.ig. Abundance (beta) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances set to 100. kappa is the negative binomial over-dispersion parameter and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are

passed as a vector of length two with the first and second elements corresponding to the lower and upper bounds of the uniform distribution. `sigma.sq.mu` are the random effect variances for any abundance random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. `tau.sq` is the residual variance for Gaussian (or zero-inflated Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and second element corresponding to the shape and scale parameters, respectively.

`tuning` a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `beta.star` (the abundance random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details. Note that no tuning is necessary for Gaussian or zero-inflated Gaussian models.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial), `'Poisson'`, `'Gaussian'`, and `'zi-Gaussian'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatially-explicit models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`save.fitted` logical value indicating whether or not fitted values and likelihood values should be saved in the resulting model object. If `save.fitted = FALSE`, the components `y.rep.samples`, `mu.samples`, and `like.samples` will not be included in the model object, and subsequent functions for calculating WAIC, fitted values, and posterior predictive checks will not work, although they all can be calculated manually if desired. Setting `save.fitted = FALSE` can be useful when working with very large data sets to minimize the amount of RAM needed when fitting and storing the model object in memory.

...	currently no additional arguments

Value

An object of class abund that is a list comprised of:

beta.samples	a coda object of posterior samples for the regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance overdispersion parameter. Only included when family = 'NB'.
tau.sq.samples	a coda object of posterior samples for the Gaussian residual variance parameter. Only included when family = 'Gaussian' or family = 'zi-Gaussian'.
y.rep.samples	a two or three-dimensional array of posterior samples for the abundance replicate (fitted) values with dimensions corresponding to MCMC samples, site, and an optional third dimension of replicate.
mu.samples	a two or three-dimensional array of posterior samples for the expected abundance samples with dimensions corresponding to MCMC samples, site, and an optional third dimension of replicate.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the model. Only included if random effects are specified in formula.
beta.star.samples	a coda object of posterior samples for the random effects. Only included if random effects are specified in formula.
like.samples	a coda object of posterior samples for the likelihood value associated with each site. Used for calculating WAIC.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Examples

```
set.seed(1010)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0, -1.5, 0.3, -0.8)
p.abund <- length(beta)
mu.RE <- list(levels = c(30),
       sigma.sq.mu = c(1.3))
kappa <- 0.5
sp <- FALSE
family <- 'NB'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re

covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2],
             abund.cov.2 = X[, , 3],
             abund.cov.3 = X[, , 4],
             abund.factor.1 = X.re[, , 1])

data.list <- list(y = y, covs = covs)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 100),
                   kappa.unif = c(0.001, 10))
# Starting values
inits.list <- list(beta = 0, kappa = kappa)

tuning <- list(kappa = 0.2, beta = 0.1, beta.star = 0.2)
n.batch <- 5
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- abund(formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3 +
                         (1 | abund.factor.1),
                         data = data.list,
                         n.batch = n.batch,
                         batch.length = batch.length,
                         inits = inits.list,
                         tuning = tuning,
                         priors = prior.list,
                         accept.rate = 0.43,
                         n.omp.threads = 1,
                         verbose = TRUE,
```

```
                        n.report = 1,
                        n.burn = n.burn,
                        n.thin = n.thin,
                        n.chains = n.chains)
summary(out)
```

`bbsData` *Count data for six warbler species in Pennsylvania, USA*

Description

Count data on 6 warblers in Pennsylvania, USA in 2018. Data come from the North American Breeding Bird Survey. Data indicate the total numer of individuals for each of 6 species counted at 50 stops along a 40km route (95 routes in the data set). The six species included in the data set are: (1) American Redstart (AMRE); (2) Blackburnian Warbler (BLBW); (3) Black-throated Blue Warbler; (4) Black-throated Green Warbler; (5) Hooded Warbler; and (6) Magnolia Warbler. Covariate data include three bioclimatic variables derived from PRISM. Two landcover variables (forest cover and developed land cover) come from USGS Earth Resources Observation and Science (EROS) Center.

Usage

```
data(bbsData)
```

Format

`bbsData` is a list with three elements:

`y`: a two-dimensional matrix of count data with rows corresponding to species (6) and columns corresponding to sites (95).

`covs`: a data frame with 95 rows and 8 columns consisting of covariates for use in modeling relative abundance.

`coords`: a numeric matrix with 95 rows and two columns containing the site coordinates. The proj4string is "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs"

Source

U.S. Geological Survey. Downloaded from https://www.sciencebase.gov/catalog/item/52b1dfa8e4b0d9b325230cd9 on August 25, 2023.

References

Daly, C., Halbleib, M., Smith, J. I., Gibson, W. P., Doggett, M. K., Taylor, G. H., Curtis, J., and Pasteris, P. P. (2008). Physiographically sensitive mapping of climatological temperature and precipitation across the conterminous united states. International Journal of Climatology: a Journal of the Royal Meteorological Society, 28(15):2031–2064

Ziolkowski Jr., D.J., Lutmerding, M., English, W.B., Aponte, V.I., and Hudson, M-A.R., 2023, North American Breeding Bird Survey Dataset 1966 - 2022: U.S. Geological Survey data release, https://doi.org/10.5066/P9GS9K64.

Sohl, T., Reker, R., Bouchard, M., Sayler, K., Dornbierer, J., Wika, S., ... & Friesz, A. (2016). Modeled historical land use and land cover for the conterminous United States. Journal of Land Use Science, 11(4), 476-499.

`bbsPredData` *Covariates and coordinates for prediction of relative warbler abundance in Pennsylvania, USA*

Description

Bioclimatic and land cover variables extracted at a 12km resolution across the state of Pennsylvania, USA for use in predicting relative abundance of six warbler species across the state. Land cover data come from USGS EROS, while climate data come from PRISM.

Usage

```
data(bbsPredData)
```

Format

`bbsPredData` is a data frame with seven columns:

`bio2`: bioclim variable 2.

`bio8`: bioclim variable 8.

`bio18`: bioclim variable 18.

`forest`: proportion of forest cover within a 5km radius.

`devel`: proportion of developed land cover within a 5km radius.

`x`: the x coordinate of the point. The proj4strig is "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs".

`y`: the y coordinate of the point. The proj4string is "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=37.5 +lon_0=-96 +x_0=0 +y_0=0 +datum=NAD83 +units=m +no_defs".

References

Daly, C., Halbleib, M., Smith, J. I., Gibson, W. P., Doggett, M. K., Taylor, G. H., Curtis, J., and Pasteris, P. P. (2008). Physiographically sensitive mapping of climatological temperature and precipitation across the conterminous united states. International Journal of Climatology: a Journal of the Royal Meteorological Society, 28(15):2031–2064

Sohl, T., Reker, R., Bouchard, M., Sayler, K., Dornbierer, J., Wika, S., ... & Friesz, A. (2016). Modeled historical land use and land cover for the conterminous United States. Journal of Land Use Science, 11(4), 476-499.

dataNMixSim *Simulated repeated count data of 6 species across 225 sites*

Description

A simulated data set of repeated count data for 6 species across 225 sites with a maximum of 3 replicate surveys performed at a given site.

Usage

```
data(dataNMixSim)
```

Format

dataNMixSim is a list with four elements:

y: a three-dimensional array of count data with dimensions of species (6), sites (225) and replicates (3).

abund.covs: a numeric matrix with 225 rows and two columns consisting of a continuous covariate and a categorical variable which may both influence abundance of the different species.

det.covs: a list of two numeric matrices with 225 rows and 3 columns. Both matrices contain a continuous covariate that may affect detection probability of the species

coords: a numeric matrix with 225 rows and two columns containing the site coordinates (X and Y). Note the data are generated across a unit square (i.e., the x and y coordinates are both between 0 and 1).

Examples

```
set.seed(6)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
# n.rep <- rep(5, J)
n.sp <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(-1, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.4, 1.2)
# Detection
alpha.mean <- c(0, 0.5, 0.8)
tau.sq.alpha <- c(0.2, 1, 1.5)
p.det <- length(alpha.mean)
# Random effects
mu.RE <- list()
mu.RE <- list(levels = c(10),
       sigma.sq.mu = c(0.5),
               beta.indx = list(1))
```

```
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
alpha.true <- alpha
sp <- TRUE
n.factors <- 3
factor.model <- TRUE
phi <- runif(n.factors, 3/1, 3 / .2)
kappa <- runif(n.sp, 0.1, 1)
family <- 'Poisson'

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
        mu.RE = mu.RE, p.RE = p.RE, sp = sp, kappa = kappa, family = family,
                factor.model = factor.model, phi = phi,
                cov.model = 'exponential', n.factors = n.factors)
table(dat$N)
apply(dat$N, 1, sum)

y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.re <- dat$X.re
X.p.re <- dat$X.p.re
coords <- dat$coords
dimnames(coords)[[2]] <- c('X', 'Y')

# Package all data into a list
abund.covs <- cbind(X, X.re)
colnames(abund.covs) <- c('int', 'abund.cov.1', 'abund.factor.1')
abund.covs <- abund.covs[, -1]
det.covs <- list(det.cov.1 = X.p[, , 2],
 det.cov.2 = X.p[, , 3])
dataNMixSim <- list(y = y,
    abund.covs = abund.covs,
    det.covs = det.covs,
                    coords = coords)
```

DS *Function for Fitting Single-Species Hierarchical Distance Sampling Models*

Description

Function for fitting single-sepcies hierarchical distance sampling models

Usage

```
DS(abund.formula, det.formula, data, inits, priors, tuning,
   n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
   transect = 'line', det.func = 'halfnormal',
   n.omp.threads = 1, verbose = TRUE,
   n.report = 100, n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
   n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, dist.breaks, and offset. y is a matrix or data frame of the observed count values, with first dimension equal to the number of sites (J) and second dimension equal to the number of distance bins. covs is a matrix or data frame containing the variables used in the the abundance and/or the detection portion of the model, with J rows for each column (variable). dist.breaks is a vector of distances that denote the breakpoints of the distance bands. dist.breaks should have length equal to the number of columns in y plus one. offset is an offset that can be used to scale estimates from abundance per transect to density per some desired unit of measure. This can be either a single value or a vector with an offset value for each site (e.g., if transects differ in length)
inits	a list with each tag corresponding to a parameter name. Valid tags are N, beta, alpha, kappa, sigma.sq.mu, and sigma.sq.p. The value portion of each tag is the parameter's initial value. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the distance sampling model, respectively. kappa is only relevant when family = 'NB'. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.normal, alpha.normal, kappa.unif, sigma.sq.mu.ig, and sigma.sq.p.ig. Abundance (beta) and detection (alpha) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances set to 100. kappa is the negative binomial dispersion

parameter and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a vector of length two with the first and second elements corresponding to the lower and upper bounds of the uniform distribution. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts/slopes or of length one if priors are the same for all random effect variances.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the number of MCMC samples in each batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`transect` the type of transect. Currently supports line transects (`'line'`) or circular transects (i.e., point counts; `'point'`).

`det.func` the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal (`'halfnormal'`) and negative exponential (`'negexp'`).

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatial models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`...` currently no additional arguments

Value

An object of class DS that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance dispersion parameter. Only included when family = 'NB'.
N.samples	a coda object of posterior samples for the latent abundance values. Note that these values always represent transect-level abundance, even when an offset is supplied.
mu.samples	a coda object of posterior samples for the latent expected abundance values. When an offset is supplied in the data object, these correspond to expected abundance per unit area (i.e., density).
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.
sigma.sq.p.samples	a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in det.formula.
beta.star.samples	a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in abund.formula.
alpha.star.samples	a coda object of posterior samples for the detection random effects. Only included if random effects are specified in det.formula.
y.rep.samples	a three-dimensional array of fitted values. Array dimensions correspond to MCMC samples, sites, and distance band.
pi.samples	a three-dimensional array of cell-specific detection probabilities. Array dimensions correspond to MCMC samples, sites, and distance band.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Examples

```
set.seed(123)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Abundance coefficients
beta <- c(1.0, 0.2, 0.3, -0.2)
p.abund <- length(beta)
# Detection coefficients
alpha <- c(-1.0, -0.3)
p.det <- length(alpha)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
sp <- FALSE
family <- 'NB'
kappa <- 0.1
offset <- 1.8
transect <- 'point'

dat <- simDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
             beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
             mu.RE = mu.RE, p.RE = p.RE, sp = sp,
             sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
             offset = offset, transect = transect)

y <- dat$y
X <- dat$X
X.re <- dat$X.re
X.p <- dat$X.p
X.p.re <- dat$X.p.re
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
```

```
                  dist.breaks = dist.breaks,
                  offset = offset)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 10),
                   alpha.normal = list(mean = 0,
                                       var = 10),
                   kappa.unif = c(0, 100))
# Starting values
inits.list <- list(alpha = 0,
                   beta = 0,
                   kappa = 1)
# Tuning values
tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.2)

out <- DS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
          det.formula = ~ det.cov.1,
          data = data.list,
          n.batch = 10,
          batch.length = 25,
          inits = inits.list,
          family = 'NB',
          det.func = 'halfnormal',
          transect = 'point',
          tuning = tuning,
          priors = prior.list,
          accept.rate = 0.43,
          n.omp.threads = 1,
          verbose = TRUE,
          n.report = 100,
          n.burn = 100,
          n.thin = 1,
          n.chains = 1)
summary(out)
```

`fitted.abund` *Extract Model Fitted Values for abund Object*

Description

Method for extracting model fitted values from a fitted GLMM (abund).

Usage

```
## S3 method for class 'abund'
fitted(object, ...)
```

Arguments

object	object of class abund.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values for fitted model objects of class abund.

Value

A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and replicates

---

fitted.DS *Extract Model Fitted Values for DS Object*

Description

Method for extracting model fitted values and cell-specific detection probabilities from a hierarchical distance sampling (DS) model.

Usage

```
## S3 method for class 'DS'
fitted(object, ...)
```

Arguments

object	object of class DS.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values and detection probabilities for fitted model objects of class DS.

Value

A list comprised of:

y.rep.samples	A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and distance bin.
pi.samples	A three-dimensional numeric array of cell-specific detection probability values. Values correspond to the probability of detecting an individual within a given distance band at a given location. Array dimensions correspond to MCMC samples, sites, and distance band.

fitted.lfMsAbund *Extract Model Fitted Values for lfMsAbund Object*

Description

Method for extracting model fitted values from a fitted latent factor multivariate GLMM (`lfMsAbund`).

Usage

```
## S3 method for class 'lfMsAbund'
fitted(object, ...)
```

Arguments

`object`	object of class `lfMsAbund`.
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values for fitted model objects of class `lfMsAbund`.

Value

A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.

fitted.lfMsDS *Extract Model Fitted Values for lfMsDS Object*

Description

Method for extracting model fitted values and cell-specific detection probabilities from a latent factor multi-species hierarchical distance sampling (`lfMsDS`) model.

Usage

```
## S3 method for class 'lfMsDS'
fitted(object, ...)
```

Arguments

`object`	object of class `lfMsDS`.
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probabilities for fitted model objects of class `lfMsDS`.

Value

A list comprised of:

`y.rep.samples`	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and distance bin.
`pi.samples`	A four-dimensional numeric array of cell-specific detection probability values. Values correspond to the probability of detecting an individual within a given distance band at a given location. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`fitted.lfMsNMix` *Extract Model Fitted Values for lfMsNMix Object*

Description

Method for extracting model fitted values and detection probability values from a fitted latent factor multi-species N-mixture (`lfMsNMix`) model.

Usage

```
## S3 method for class 'lfMsNMix'
fitted(object, type = 'marginal', ...)
```

Arguments

`object`	object of class `lfMsNMix`.
`type`	a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (`type = 'conditional'`) estimated during the model or based on the marginal expected abundance values (`type = 'marginal'`).
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probability values for fitted model objects of class `lfMsNMix`.

Value

A list comprised of:

`y.rep.samples`	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.
`p.samples`	A four-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, species, sites, and replicates.

`fitted.msAbund` *Extract Model Fitted Values for msAbund Object*

Description

Method for extracting model fitted values from a fitted multivariate GLMM (`msAbund`).

Usage

```
## S3 method for class 'msAbund'
fitted(object, ...)
```

Arguments

`object`	object of class `msAbund`.
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values for fitted model objects of class `msAbund`.

Value

A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.

fitted.msDS *Extract Model Fitted Values for msDS Object*

Description

Method for extracting model fitted values and cell-specific detection probabilities from a multi-species hierarchical distance sampling (msDS) model.

Usage

```
## S3 method for class 'msDS'
fitted(object, ...)
```

Arguments

object	object of class msDS.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values and detection probabilities for fitted model objects of class msDS.

Value

A list comprised of:

y.rep.samples	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and distance bin.
pi.samples	A four-dimensional numeric array of cell-specific detection probability values. Values correspond to the probability of detecting an individual within a given distance band at a given location. Array dimensions correspond to MCMC samples, species, sites, and distance band.

fitted.msNMix *Extract Model Fitted Values for msNMix Object*

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species N-mixture (msNMix) model.

Usage

```
## S3 method for class 'msNMix'
fitted(object, type = 'marginal', ...)
```

Arguments

object	object of class msNMix.
type	a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (type = 'conditional') estimated during the model or based on the marginal expected abundance values (type = 'marginal').
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values and detection probability values for fitted model objects of class msNMix.

Value

A list comprised of:

y.rep.samples	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.
p.samples	A four-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, species, sites, and replicates.

---

fitted.NMix *Extract Model Fitted Values for NMix Object*

Description

Method for extracting model fitted values and detection probabilities from a fitted N-mixture (NMix) model.

Usage

```
## S3 method for class 'NMix'
fitted(object, type = 'marginal', ...)
```

Arguments

object	object of class NMix.
type	a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (type = 'conditional') estimated during the model or based on the marginal expected abundance values (type = 'marginal').
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values and detection probabilities for fitted model objects of class NMix.

Value

A list comprised of:

y.rep.samples	A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and replicates.
p.samples	A three-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, sites, and replicates.

---

fitted.sfMsAbund *Extract Model Fitted Values for sfMsAbund Object*

Description

Method for extracting model fitted values from a fitted spatial factor multivariate GLMM (sfMsAbund).

Usage

```
## S3 method for class 'sfMsAbund'
fitted(object, ...)
```

Arguments

object	object of class sfMsAbund.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values for fitted model objects of class sfMsAbund.

Value

A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.

`fitted.sfMsDS` *Extract Model Fitted Values for sfMsDS Object*

Description

Method for extracting model fitted values and cell-specific detection probabilities from a spatial factor multi-species hierarchical distance sampling (`sfMsDS`) model.

Usage

```
## S3 method for class 'sfMsDS'
fitted(object, ...)
```

Arguments

`object`	object of class `sfMsDS`.
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probabilities for fitted model objects of class `sfMsDS`.

Value

A list comprised of:

`y.rep.samples`	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and distance bin.
`pi.samples`	A four-dimensional numeric array of cell-specific detection probability values. Values correspond to the probability of detecting an individual within a given distance band at a given location. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`fitted.sfMsNMix` *Extract Model Fitted Values for sfMsNMix Object*

Description

Method for extracting model fitted values and detection probability values from a fitted spatial factor multi-species N-mixture (`sfMsNMix`) model.

Usage

```
## S3 method for class 'sfMsNMix'
fitted(object, type = 'marginal', ...)
```

Arguments

`object`	object of class `sfMsNMix`.
`type`	a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (`type = 'conditional'`) estimated during the model or based on the marginal expected abundance values (`type = 'marginal'`).
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probability values for fitted model objects of class `sfMsNMix`.

Value

A list comprised of:

`y.rep.samples`	A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.
`p.samples`	A four-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, species, sites, and replicates.

fitted.spAbund *Extract Model Fitted Values for spAbund Object*

Description

Method for extracting model fitted values from a fitted spatial GLMM (spAbund).

Usage

```
## S3 method for class 'spAbund'
fitted(object, ...)
```

Arguments

object	object of class spAbund.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values for fitted model objects of class spAbund.

Value

A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and replicates

fitted.spDS *Extract Model Fitted Values for spDS Object*

Description

Method for extracting model fitted values and cell-specific detection probabilities from a spatial hierarchical distance sampling (spDS) model.

Usage

```
## S3 method for class 'spDS'
fitted(object, ...)
```

Arguments

object	object of class spDS.
...	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probabilities for fitted model objects of class `spDS`.

Value

A list comprised of:

`y.rep.samples`	A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and distance bin.
`pi.samples`	A three-dimensional numeric array of cell-specific detection probability values. Values correspond to the probability of detecting an individual within a given distance band at a given location. Array dimensions correspond to MCMC samples, sites, and distance band.

`fitted.spNMix` *Extract Model Fitted Values for spNMix Object*

Description

Method for extracting model fitted values and detection probabilities from a fitted spatial N-mixture (`spNMix`) model.

Usage

```
## S3 method for class 'spNMix'
fitted(object, type = 'marginal', ...)
```

Arguments

`object`	object of class `spNMix`.
`type`	a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (`type = 'conditional'`) estimated during the model or based on the marginal expected abundance values (`type = 'marginal'`).
`...`	currently no additional arguments

Details

A method to the generic `fitted` function to extract fitted values and detection probabilities for fitted model objects of class `spNMix`.

Value

A list comprised of:

y.rep.samples	A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and replicates.
p.samples	A three-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, sites, and replicates.

---

fitted.svcAbund *Extract Model Fitted Values for svcAbund Object*

Description

Method for extracting model fitted values from a fitted spatially-varying coefficient GLMM (svcAbund).

Usage

```
## S3 method for class 'svcAbund'
fitted(object, ...)
```

Arguments

object	object of class svcAbund.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values for fitted model objects of class svcAbund.

Value

A three-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, sites, and replicates

fitted.svcMsAbund *Extract Model Fitted Values for svcMsAbund Object*

Description

Method for extracting model fitted values from a fitted multivatiate spatially-varying coefficient GLMM (svcMsAbund).

Usage

```
## S3 method for class 'svcMsAbund'
fitted(object, ...)
```

Arguments

object	object of class svcMsAbund.
...	currently no additional arguments

Details

A method to the generic fitted function to extract fitted values for fitted model objects of class svcMsAbund.

Value

A four-dimensional numeric array of fitted values for use in Goodness of Fit assessments. Array dimensions correspond to MCMC samples, species, sites, and replicates.

hbefCount2015 *Count data of 12 foliage gleaning bird species in 2015 in the Hubbard Brook Experimental Forest*

Description

Repeated count data of 12 foliage gleaning bird species in 2015 in the Hubbard Brook Experimental Forest (HBEF) in New Hampshire, USA. Data were collected at 373 sites over three replicate point counts each of 10 minutes in length, with a detection radius of 100m. Some sites were not visited for all three replicates. The 12 species included in the data set are as follows: (1) AMRE: American Redstart; (2) BAWW: Black-and-white Warbler; (3) BHVI: Blue-headed Vireo; (4) BLBW: Blackburnian Warbler; (5) BLPW: Blackpoll Warbler; (6) BTBW: Black-throated Blue Warbler; (7) BTNW: BLack-throated Green Warbler; (8) CAWA: Canada Warbler; (9) MAWA: Magnolia Warbler; (10) NAWA: Nashville Warbler; (11) OVEN: Ovenbird; (12) REVI: Red-eyed Vireo.

Usage

```
data(hbefCount2015)
```

Format

hbefCount2015 is a list with four elements:

y: a three-dimensional array of count data with dimensions of species (12), sites (373) and replicates (3).

abund.covs: a data frame with 373 rows and one column consisting of the elevation at each site.

det.covs: a list of two numeric matrices with 373 rows and 3 columns. The first element is the day of year when the survey was conducted for a given site and replicate. The second element is the time of day when the survey was conducted.

coords: a numeric matrix with 373 rows and two columns containing the site coordinates (Easting and Northing) in UTM Zone 19. The proj4string is "+proj=utm +zone=19 +units=m +datum=NAD83".

Source

Rodenhouse, N. and S. Sillett. 2019. Valleywide Bird Survey, Hubbard Brook Experimental Forest, 1999-2016 (ongoing) ver 3. Environmental Data Initiative. doi:10.6073/pasta/faca2b2cf2db9d415c39b695cc7fc217 (Accessed 2021-09-07)

References

Doser, J. W., Leuenberger, W., Sillett, T. S., Hallworth, M. T. & Zipkin, E. F. (2022). Integrated community occupancy models: A framework to assess occurrence and biodiversity dynamics using multiple data sources. Methods in Ecology and Evolution, 00, 1-14. doi:10.1111/2041210X.13811

---

lfMsAbund *Function for Fitting Latent Factor Multivariate Abundance GLMMs*

Description

Function for fitting multivariate generalized linear (mixed) models with species correlations (i.e., an abundance-based joint species distribution model). We use a factor modeling approach for dimension reduction.

Usage

```
lfMsAbund(formula, data, inits, priors, tuning, n.factors,
          n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
          n.omp.threads = 1, verbose = TRUE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1, n.chains = 1,
          save.fitted = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, z, and offset. y is a two or three-dimensional array of observed count data. The first dimension of the array is equal to the number of species and the second dimension is equal to the number of sites. If specified as a three-dimensional array, the third dimension corresponds to replicate observations at each site (e.g., subsamples, repeated sampling over multiple seasons). covs is a list or data frame containing the variables used in the model. If a data frame, each row of covs is a site and each column is a variable. If specified as a list, each list element is a different covariate, which can be site-level or observation-level. Site-level covariates are specified as a vector of length J, while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicate observations at a given site. coords is a matrix or data frame with two columns that contain the spatial coordinates of each site. Note that spAbundance assumes coordinates are specified in a projected coordinate system. For zi-Gaussian models, the tag z is used to specify the binary component of the zi-Gaussian model and should have the same dimensions as y. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value, a vector with an offset for each site (e.g., if survey area differed in size), or a site x replicate matrix if more than one count is available at a given site.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta.comm, beta, tau.sq.beta, sigma.sq.mu, kappa, lambda, w, tau.sq. kappa is only specified if family = 'NB', tau.sq is only specified for Gaussian and zi-Gaussian models, and sigma.sq.mu is only specified if random effects are included in formula. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, tau.sq.beta.ig, sigma.sq.mu, kappa.unif, tau.sq.ig. Community-level (beta.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances to 100. Community-level variance parameters (tau.sq.beta) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. sigma.sq.mu are the random effect variances

random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts or of length one if priors are the same for all random effect variances. `kappa` is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters. `tau.sq` is the species-specific residual variance for Gaussian (or zi-Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a list of length two, with the first and second element corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of species or a single value if priors are the same for all species.

`tuning` a list with each tag corresponding to a parameter name, whose whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `beta.star` (the abundance random effect values), `kappa`, `lambda` (the latent factor loadings), and `w` (the latent factors). See Roberts and Rosenthal (2009) for details. Note that no tuning is necessary for Gaussian or zi-Gaussian models.

`n.factors` the number of factors to use in the latent factor model approach. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and N (the number of species in the community).

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for adaptive MCMC. Defaul is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial), `'Poisson'`, `'Gaussian'`, and `'zi-Gaussian'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report Metropolis sampler acceptance and MCMC progress. Note this is specified in terms of batches and not overall samples for spatial models.

n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
save.fitted	logical value indicating whether or not fitted values and likelihood values should be saved in the resulting model object. If save.fitted = FALSE, the components y.rep.samples, mu.samples, and like.samples will not be included in the model object, and subsequent functions for calculating WAIC, fitted values, and posterior predictive checks will not work, although they all can be calculated manually if desired. Setting save.fitted = FALSE can be useful when working with very large data sets to minimize the amount of RAM needed when fitting and storing the model object in memory.
...	currently no additional arguments

Value

An object of class lfMsAbund that is a list comprised of:

beta.comm.samples	a coda object of posterior samples for the community level regression coefficients.
tau.sq.beta.samples	a coda object of posterior samples for the abundance community variance parameters.
beta.samples	a coda object of posterior samples for the species level abundance regression coefficients.
kappa.samples	a coda object of posterior samples for the species level abundance dispersion parameters. Only included when family = 'NB'.
tau.sq.samples	a coda object of posterior samples for the Gaussian residual variance parameter. Only included when family = 'Gaussian' or family = 'zi-Gaussian'.
lambda.samples	a coda object of posterior samples for the latent factor loadings.
w.samples	a three-dimensional array of posterior samples for the latent effects for each latent factor. Array dimensions correspond to MCMC sample, latent factor, then site.
y.rep.samples	a three or four-dimensional array of posterior samples for the fitted (replicate) values for each species with dimensions corresponding to MCMC sample, species, site, and replicate.
mu.samples	a three or four-dimensional array of posterior samples for the expected abundance values for each species with dimensions corresponding to MCMC samples, species, site, and replicate.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`like.samples`
: a three-dimensional array of posterior samples for the likelihood value associated with each site and species. Used for calculating WAIC.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

References

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Random effects (two random intercepts)
mu.RE <- list(levels = c(10, 15),
              sigma.sq.mu = c(0.43, 0.5))
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- FALSE
kappa <- runif(n.sp, 0.1, 1)
factor.model <- TRUE
n.factors <- 3
```

```
dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep,
                  n.sp = n.sp, beta = beta, mu.RE = mu.RE,
                  sp = sp, kappa = kappa, family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re
coords <- dat$coords

# Package all data into a list
covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2],
             abund.factor.1 = X.re[, , 1],
             abund.factor.2 = X.re[, , 2])
data.list <- list(y = y, covs = covs, coords = coords)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0, beta = 0, kappa = 0.5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1, beta.star = 0.1,
                    lambda = 0.5, w = 0.5)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- lfMsAbund(formula = ~ abund.cov.1 + (1 | abund.factor.1) +
                             (1 | abund.factor.2),
                 data = data.list,
                 n.batch = n.batch,
                 inits = inits.list,
                 priors = prior.list,
                 tuning = tuning.list,
                 batch.length = batch.length,
                 n.factors = n.factors,
                 n.omp.threads = 3,
                 verbose = TRUE,
                 n.report = 1,
                 n.burn = n.burn,
                 n.thin = n.thin,
                 n.chains = n.chains)
summary(out)
```

`lfMsDS` *Function for Fitting Latent Factor Multi-Species Hierarchical Distance Sampling Models*

Description

Function for fitting latent factor multi-species hierarchical distance sampling models.

Usage

```
lfMsDS(abund.formula, det.formula, data, inits, priors,
       tuning, n.factors, n.batch, batch.length, accept.rate = 0.43,
       family = 'Poisson', transect = 'line', det.func = 'halfnormal',
       n.omp.threads = 1, verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
       n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, dist.breaks, and offset. y is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. covs is a matrix or data frame containing the variables used in the abundance and/or the detection portion of the model, with J rows for each column (variable). dist.breaks is a vector of distances that denote the breakpoints of the distance bands. dist.breaks should have length equal to the third dimension of y plus one. offset is an offset that can be used to scale estimates from abundance per transect to density per some desired unit of measure. This can be either a single value or a vector with an offset value for each site (e.g., if transects differ in length). coords is a matrix or data frame with two columns that contain the spatial coordinates of each site. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
inits	a list with each tag corresponding to a parameter name. Valid tags are alpha.comm, beta.comm, beta, alpha, tau.sq.beta, tau.sq.alpha, sigma.sq.mu, sigma.sq.p, kappa, N, lambda, w. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the model, respectively. kappa is only relevant when family = 'NB'. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.

priors — a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, alpha.comm.normal, tau.sq.beta.ig, tau.sq.alpha.ig, sigma.sq.mu.ig, sigma.sq.p.ig, and kappa.unif. Community-level abundance (beta.comm) and detection (alpha.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances are set to 100. Community-level variance parameters for abundance (tau.sq.beta) and detection (tau.sq.alpha) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. sigma.sq.mu and sigma.sq.p are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters.

tuning — a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are beta, alpha, lambda (the latent factor loadings), w (the latent factors), beta.star (the abundance random effect values), alpha.star (the detection random effect values), and kappa. See Roberts and Rosenthal (2009) for details.

n.factors — the number of factors to use in the latent factor model approach. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and N (the number of species in the community).

n.batch — the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

batch.length — the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

accept.rate — target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
transect	the type of transect. Currently supports line transects ('line') or circular transects (i.e., point counts; 'point').
det.func	the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal ('halfnormal') and negative exponential ('negexp').
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hypterthreaded cores. Note, n.omp.threads > 1 might not work on some systems. Currently only relevant for spatial models.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report MCMC progress.
n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class lfMsDS that is a list comprised of:

beta.comm.samples	a coda object of posterior samples for the community level abundance regression coefficients.
alpha.comm.samples	a coda object of posterior samples for the community level detection regression coefficients.
tau.sq.beta.samples	a coda object of posterior samples for the abundance community variance parameters.
tau.sq.alpha.samples	a coda object of posterior samples for the detection community variance parameters.
beta.samples	a coda object of posterior samples for the species level abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the species level detection regression coefficients.
kappa.samples	a coda object of posterior samples for the species level abundance dispersion parameters. Only included when family = 'NB'.

`lambda.samples` a coda object of posterior samples for the latent factor loadings.

`w.samples` a three-dimensional array of posterior samples for the latent effects for each latent factor.

`N.samples` a three-dimensional array of posterior samples for the latent abundance values for each species. Note that these values always represent transect-level abundance, even when an offset is supplied. Array dimensions correspond to MCMC sample, species, and site.

`mu.samples` a three-dimensional array of posterior samples for the latent expected abundance values for each species. When an offset is supplied in the `data` object, these correspond to expected abundance per unit area (i.e., density). Array dimensions correspond to MCMC sample, species, and site.

`sigma.sq.mu.samples`
a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in `abund.formula`.

`sigma.sq.p.samples`
a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples`
a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples`
a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`y.rep.samples` a four-dimensional array of fitted values. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`pi.samples` a four-dimensional array of cell-specific detection probabilities. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`rhat` a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS` a list of effective sample sizes for some of the model parameters.

`run.time` MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Sollmann, R., Gardner, B., Williams, K. A., Gilbert, A. T., & Veit, R. R. (2016). A hierarchical distance sampling model to estimate abundance and covariate associations of species and communities. Methods in Ecology and Evolution, 7(5), 529-537.

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE
family <- 'Poisson'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
factor.model <- TRUE
n.factors <- 3

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
                 mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
                 sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
                 offset = offset, transect = transect, factor.model = factor.model,
```

```
                n.factors = n.factors)

y <- dat$y
X <- dat$X
X.p <- dat$X.p
coords <- dat$coords
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
                   alpha.comm.normal = list(mean = 0, var = 10),
                   kappa.unif = list(0, 100),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, kappa = 1)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8, lambda = 1, w = 1)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- lfMsDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
      det.formula = ~ det.cov.1,
      data = data.list,
      n.batch = n.batch,
      batch.length = batch.length,
      inits = inits.list,
      family = 'Poisson',
      det.func = 'halfnormal',
      transect = transect,
      tuning = tuning,
      n.factors = n.factors,
      priors = prior.list,
      accept.rate = 0.43,
      n.omp.threads = 1,
      verbose = TRUE,
      n.report = 10,
```

```
      n.burn = n.burn,
      n.thin = n.thin,
      n.chains = n.chains)
summary(out, level = 'community')
```

`lfMsNMix` *Function for Fitting Latent Factor Multi-species N-mixture Models*

Description

Function for fitting multi-species N-mixture models with species correlations (i.e., an abundance-based joint species distribution model with imperfect detection). We use a factor modeling approach for dimension reduction.

Usage

```
lfMsNMix(abund.formula, det.formula, data, inits, priors,
         tuning, n.factors, n.batch, batch.length, accept.rate = 0.43,
         family = 'Poisson', n.omp.threads = 1, verbose = TRUE, n.report = 100,
         n.burn = round(.10 * n.samples), n.thin = 1,
         n.chains = 1, ...)
```

Arguments

`abund.formula`	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
`det.formula`	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
`data`	a list containing data necessary for model fitting. Valid tags are `y`, `abund.covs`, `det.covs`, `coords`, and `offset`. `y` is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. `abund.covs` is a matrix or data frame containing the variables used in the abundance portion of the model, with J rows for each column (variable). `det.covs` is a list of variables included in the detection portion of the model. Each list element is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector of length J while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicates at a given site. `coords` is a matrix or data frame with two columns that contain the spatial coordinates of each site. Note that `spAbundance` assumes coordinates are specified in a projected coordinate system. `offset` is an offset to use in the abundance model (e.g., an area offset).

This can be either a single value or a vector with an offset for each site (e.g., if survey area differed in size).

`inits` a list with each tag corresponding to a parameter name. Valid tags are `alpha.comm`, `beta.comm`, `beta`, `alpha`, `tau.sq.beta`, `tau.sq.alpha`, `sigma.sq.mu`, `sigma.sq.p`, `lambda`, `w`, `kappa`, and `N`. `sigma.sq.mu` and `sigma.sq.p` are only relevant when including random effects in the abundance and detection portion of the model, respectively. `kappa` is only relevant when `family = 'NB'`. The value portion of each tag is the parameter's initial value. See `priors` description for definition of each parameter name. Additionally, the tag `fix` can be set to `TRUE` to fix the starting values across all chains. If `fix` is not specified (the default), starting values are varied randomly across chains.

`priors` a list with each tag corresponding to a parameter name. Valid tags are `beta.comm.normal`, `alpha.comm.normal`, `tau.sq.beta.ig`, `tau.sq.alpha.ig`, `sigma.sq.mu.ig`, `sigma.sq.p.ig`, and `kappa.unif`. Community-level abundance (`beta.comm`) and detection (`alpha.comm`) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances for the abundance coefficients are set to 100 and for the detection coefficients are set to 2.72. Community-level variance parameters for abundance (`tau.sq.beta`) and detection (`tau.sq.alpha`) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. `kappa` is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters.

`tuning` a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), `lambda` (the latent factor loadings), `w` (the latent factors), and `kappa`. See Roberts and Rosenthal (2009) for details.

`n.factors` the number of factors to use in the latent factor model approach. Typically, the

number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and N (the number of species in the community).

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatial models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`...` currently no additional arguments

Value

An object of class `lfMsNMix` that is a list comprised of:

`beta.comm.samples`
a coda object of posterior samples for the community level abundance regression coefficients.

`alpha.comm.samples`
a coda object of posterior samples for the community level detection regression coefficients.

`tau.sq.beta.samples`
a coda object of posterior samples for the abundance community variance parameters.

`tau.sq.alpha.samples`
a coda object of posterior samples for the detection community variance parameters.

`beta.samples` a coda object of posterior samples for the species level abundance regression coefficients.

`alpha.samples` a coda object of posterior samples for the species level detection regression coefficients.

`lambda.samples` a coda object of posterior samples for the latent factor loadings.

`w.samples` a three-dimensional array of posterior samples for the latent effects for each latent factor.

`kappa.samples` a coda object of posterior samples for the species level abundance dispersion parameters. Only included when `family = 'NB'`.

`N.samples` a three-dimensional array of posterior samples for the latent abundance values for each species.

`mu.samples` a three-dimensional array of posterior samples for the latent expected abundance values for each species.

`sigma.sq.mu.samples` a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in `abund.formula`.

`sigma.sq.p.samples` a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples` a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples` a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`rhat` a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS` a list of effective sample sizes for some of the model parameters.

`run.time` MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation. Note that detection probability estimated values are not included in the model object, but can be extracted using `fitted()`.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Yamaura, Y., Royle, J. A., Shimada, N., Asanuma, S., Sato, T., Taki, H., & Makino, S. I. (2012). Biodiversity of man-made open habitats in an underused country: a class of multispecies abundance models for count data. Biodiversity and Conservation, 21(6), 1365-1380.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(5, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
# Abundance
beta.mean <- c(0, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Detection
alpha.mean <- c(0, 0.5, 0.8)
tau.sq.alpha <- c(0.2, 1, 1.5)
p.det <- length(alpha.mean)
# Random effects
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 3

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
                 mu.RE = mu.RE, p.RE = p.RE, sp = FALSE, family = 'Poisson',
                 factor.model = TRUE, n.factors = n.factors)

y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.re <- dat$X.re
X.p.re <- dat$X.p.re
coords <- dat$coords

# Package all data into a list
abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')
det.covs <- list(det.cov.1 = as.data.frame(X.p[, , 2]),
                 det.cov.2 = as.data.frame(X.p[, , 3]))
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)
prior.list <- list(beta.comm.normal = list(mean = rep(0, p.abund),
                                           var = rep(100, p.abund)),
```

```
                    alpha.comm.normal = list(mean = rep(0, p.det),
                                             var = rep(2.72, p.det)),
                    tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                    tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
inits.list <- list(beta.comm = 0, alpha.comm = 0,
                    beta = 0, alpha = 0,
                    tau.sq.beta = 0.5, tau.sq.alpha = 0.5,
                    N = apply(y, c(1, 2), max, na.rm = TRUE))
tuning.list <- list(beta = 0.5, alpha = 0.5, lambda = 0.5, w = 0.5)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- lfMsNMix(abund.formula = ~ abund.cov.1,
                det.formula = ~ det.cov.1 + det.cov.2,
                data = data.list,
                n.batch = n.batch,
                inits = inits.list,
                priors = prior.list,
                tuning = tuning.list,
                batch.length = batch.length,
                n.omp.threads = 1,
                n.factors = n.factors,
                verbose = TRUE,
                n.report = 1,
                n.burn = n.burn,
                n.thin = n.thin,
                n.chains = n.chains)

summary(out, level = 'community')
```

---

msAbund *Function for Fitting Multivariate Abundance GLMMs*

Description

The function msAbund fits multivariate abundance GLMMs.

Usage

```
msAbund(formula, data, inits, priors, tuning,
        n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
        n.omp.threads = 1, verbose = TRUE, n.report = 100,
        n.burn = round(.10 * n.batch * batch.length), n.thin = 1, n.chains = 1,
        save.fitted = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are `y`, `covs`, `z`, and `offset`. `y` is a two or three-dimensional array of observed count data. The first dimension of the array is equal to the number of species and the second dimension is equal to the number of sites. If specified as a three-dimensional array, the third dimension corresponds to replicate observations at each site (e.g., subsamples, repeated sampling over multiple seasons). `covs` is a list or data frame containing the variables used in the model. If a data frame, each row of `covs` is a site and each column is a variable. If a list, each list element is a different covariate, which can be site-level or observation-level. Site-level covariates are specified as a vector of length J, while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicate observations at a given site. For zero-inflated Gaussian models, the tag `z` is used to specify the binary component of the model and should have the same dimensions as `y`. `offset` is an offset to use in the abundance model (e.g., an area offset). This can be either a single value, a vector with an offset for each site (e.g., if survey area differed in size), or a site x replicate matrix if more than one count is available at a given site.
inits	a list with each tag corresponding to a parameter name. Valid tags are `beta.comm`, `beta`, `tau.sq.beta`, `sigma.sq.mu`, `kappa`, `tau.sq`. `kappa` is only specified if `family = 'NB'`, `tau.sq` is only specified for Gaussian or zero-inflated Gaussian models, and `sigma.sq.mu` is only specified if random effects are included in `formula`. The value portion of each tag is the parameter's initial value. See `priors` description for definition of each parameter name. Additionally, the tag `fix` can be set to `TRUE` to fix the starting values across all chains. If `fix` is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are `beta.comm.normal`, `tau.sq.beta.ig`, `sigma.sq.mu`, `kappa.unif`, `tau.sq.ig`. Community-level (`beta.comm`) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances to 100. Community-level variance parameters (`tau.sq.beta`) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. `sigma.sq.mu` are the random effect variances random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as

a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts or of length one if priors are the same for all random effect variances. `kappa` is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters. `tau.sq` is the species-specific residual variance for Gaussian (or zero-inflated Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a list of length two, with the first and second element corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of species or a single value if priors are the same for all species.

`tuning` a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `beta.star` (the abundance random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details. Note that no tuning is necessary for Gaussian or zero-inflated Gaussian models.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for adaptive MCMC. Defaul is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial), `'Poisson'`, `'Gaussian'`, and `'zi-Gaussian'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report Metropolis sampler acceptance and MCMC progress. Note this is specified in terms of batches and not overall samples for spatial models.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`save.fitted` logical value indicating whether or not fitted values and likelihood values should be saved in the resulting model object. If `save.fitted = FALSE`, the components `y.rep.samples`, `mu.samples`, and `like.samples` will not be included in

the model object, and subsequent functions for calculating WAIC, fitted values, and posterior predictive checks will not work, although they all can be calculated manually if desired. Setting `save.fitted = FALSE` can be useful when working with very large data sets to minimize the amount of RAM needed when fitting and storing the model object in memory.

`...` currently no additional arguments

Value

An object of class `msAbund` that is a list comprised of:

`beta.comm.samples`
: a `coda` object of posterior samples for the community level regression coefficients.

`tau.sq.beta.samples`
: a `coda` object of posterior samples for the abundance community variance parameters.

`beta.samples` a `coda` object of posterior samples for the species level abundance regression coefficients.

`kappa.samples` a `coda` object of posterior samples for the species level abundance dispersion parameters. Only included when `family = 'NB'`.

`tau.sq.samples` a `coda` object of posterior samples for the Gaussian residual variance parameter. Only included when `family = 'Gaussian'` or `family = 'zi-Gaussian'`.

`y.rep.samples` a three or four-dimensional array of posterior samples for the fitted (replicate) values for each species with dimensions corresponding to MCMC sample, species, site, and replicate.

`mu.samples` a three or four-dimensional array of posterior samples for the expected abundance values for each species with dimensions corresponding to MCMC samples, species, site, and replicate.

`sigma.sq.mu.samples`
: a `coda` object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in `abund.formula`.

`beta.star.samples`
: a `coda` object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`like.samples` a three-dimensional array of posterior samples for the likelihood value associated with each site and species. Used for calculating WAIC.

`rhat` a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS` a list of effective sample sizes for some of the model parameters.

`run.time` MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

References

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Random effects (two random intercepts)
mu.RE <- list(levels = c(10, 15),
              sigma.sq.mu = c(0.43, 0.5))
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- FALSE
kappa <- runif(n.sp, 0.1, 1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re

# Package all data into a list
covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2],
             abund.factor.1 = X.re[, , 1],
             abund.factor.2 = X.re[, , 2])
data.list <- list(y = y, covs = covs)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   tau.sq.beta.ig = list(a = .1, b = .1))
```

```
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   kappa = 0.5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1, beta.star = 0.1)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- msAbund(formula = ~ abund.cov.1 + (1 | abund.factor.1) +
                           (1 | abund.factor.2),
               data = data.list,
               n.batch = n.batch,
               inits = inits.list,
               priors = prior.list,
               tuning = tuning.list,
               batch.length = batch.length,
               n.omp.threads = 3,
               verbose = TRUE,
               n.report = 1,
               n.burn = n.burn,
               n.thin = n.thin,
               n.chains = n.chains)
summary(out)
```

`msDS` *Function for Fitting Multi-Species Hierarchical Distance Sampling Models*

Description

Function for fitting multi-species hierarchical distance sampling models.

Usage

```
msDS(abund.formula, det.formula, data, inits, priors,
     tuning, n.batch, batch.length, accept.rate = 0.43,
     family = 'Poisson', transect = 'line', det.func = 'halfnormal',
     n.omp.threads = 1, verbose = TRUE, n.report = 100,
     n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
     n.chains = 1, ...)
```

Arguments

`abund.formula`	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See

example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).

`det.formula` a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).

`data` a list containing data necessary for model fitting. Valid tags are `y`, `covs`, and `dist.breaks`, and `offset`. `y` is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. `covs` is a matrix or data frame containing the variables used in the abundance and/or the detection portion of the model, with J rows for each column (variable). `dist.breaks` is a vector of distances that denote the breakpoints of the distance bands. `dist.breaks` should have length equal to the third dimension of `y` plus one. `offset` is an offset that can be used to scale estimates from abundance per transect to density per some desired unit of measure. This can be either a single value or a vector with an offset value for each site (e.g., if transects differ in length)

`inits` a list with each tag corresponding to a parameter name. Valid tags are `alpha.comm`, `beta.comm`, `beta`, `alpha`, `tau.sq.beta`, `tau.sq.alpha`, `sigma.sq.mu`, `sigma.sq.p`, `kappa`, and `N`. `sigma.sq.mu` and `sigma.sq.p` are only relevant when including random effects in the abundance and detection portion of the model, respectively. `kappa` is only relevant when `family = 'NB'`. The value portion of each tag is the parameter's initial value. See `priors` description for definition of each parameter name. Additionally, the tag `fix` can be set to `TRUE` to fix the starting values across all chains. If `fix` is not specified (the default), starting values are varied randomly across chains.

`priors` a list with each tag corresponding to a parameter name. Valid tags are `beta.comm.normal`, `alpha.comm.normal`, `tau.sq.beta.ig`, `tau.sq.alpha.ig`, `sigma.sq.mu.ig`, `sigma.sq.p.ig`, and `kappa.unif`. Community-level abundance (`beta.comm`) and detection (`alpha.comm`) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances are set to 100. Community-level variance parameters for abundance (`tau.sq.beta`) and detection (`tau.sq.alpha`) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length

two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters.

`tuning` a list with each tag corresponding to a parameter name, whose whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details.

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`transect` the type of transect. Currently supports line transects (`'line'`) or circular transects (i.e., point counts; `'point'`).

`det.func` the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal (`'halfnormal'`) and negative exponential (`'negexp'`).

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatial models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`...` currently no additional arguments

Value

An object of class msDS that is a list comprised of:

beta.comm.samples
: a coda object of posterior samples for the community level abundance regression coefficients.

alpha.comm.samples
: a coda object of posterior samples for the community level detection regression coefficients.

tau.sq.beta.samples
: a coda object of posterior samples for the abundance community variance parameters.

tau.sq.alpha.samples
: a coda object of posterior samples for the detection community variance parameters.

beta.samples
: a coda object of posterior samples for the species level abundance regression coefficients.

alpha.samples
: a coda object of posterior samples for the species level detection regression coefficients.

kappa.samples
: a coda object of posterior samples for the species level abundance dispersion parameters. Only included when family = 'NB'.

N.samples
: a three-dimensional array of posterior samples for the latent abundance values for each species. Note that these values always represent transect-level abundance, even when an offset is supplied. Array dimensions correspond to MCMC sample, species, and site.

mu.samples
: a three-dimensional array of posterior samples for the latent expected abundance values for each species. When an offset is supplied in the data object, these correspond to expected abundance per unit area (i.e., density). Array dimensions correspond to MCMC samples, species, and site.

sigma.sq.mu.samples
: a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.

sigma.sq.p.samples
: a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in det.formula.

beta.star.samples
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in abund.formula.

alpha.star.samples
: a coda object of posterior samples for the detection random effects. Only included if random effects are specified in det.formula.

y.rep.samples
: a four-dimensional array of fitted values. Array dimensions correspond to MCMC samples, species, sites, and distance band.

pi.samples	a four-dimensional array of cell-specific detection probabilities. Array dimensions correspond to MCMC samples, species, sites, and distance band.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	MCMC sampler execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Sollmann, R., Gardner, B., Williams, K. A., Gilbert, A. T., & Veit, R. R. (2016). A hierarchical distance sampling model to estimate abundance and covariate associations of species and communities. Methods in Ecology and Evolution, 7(5), 529-537.

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
```

```
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE
family <- 'Poisson'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
factor.model <- FALSE

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func,
               mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
               sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
               offset = offset, transect = transect, factor.model = factor.model)

y <- dat$y
X <- dat$X
X.re <- dat$X.re
X.p <- dat$X.p
X.p.re <- dat$X.p.re
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
                   alpha.comm.normal = list(mean = 0,
                                            var = 10),
                   kappa.unif = list(0, 100),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, kappa = 1)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8)

n.batch <- 4
batch.length <- 25
n.burn <- 0
```

```
n.thin <- 1
n.chains <- 1

out <- msDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
            det.formula = ~ det.cov.1,
            data = data.list,
            n.batch = n.batch,
            batch.length = batch.length,
            inits = inits.list,
            family = 'Poisson',
            det.func = 'halfnormal',
            transect = transect,
            tuning = tuning,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 10,
            n.burn = n.burn,
            n.thin = n.thin,
            n.chains = n.chains)
summary(out, level = 'community')
```

`msNMix` *Function for Fitting Multi-species N-mixture Models*

Description

Function for fitting multi-species N-mixture models.

Usage

```
msNMix(abund.formula, det.formula, data, inits, priors,
       tuning, n.batch, batch.length, accept.rate = 0.43,
       family = 'Poisson', n.omp.threads = 1, verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.samples), n.thin = 1,
       n.chains = 1, ...)
```

Arguments

`abund.formula`	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
`det.formula`	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).

data — a list containing data necessary for model fitting. Valid tags are y, abund.covs, det.covs, and offset. y is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. abund.covs is a matrix or data frame containing the variables used in the abundance portion of the model, with J rows for each column (variable). det.covs is a list of variables included in the detection portion of the model. Each list element is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector of length J while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicates at a given site. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value or a vector with an offset for each site (e.g., if survey area differed in size).

inits — a list with each tag corresponding to a parameter name. Valid tags are alpha.comm, beta.comm, beta, alpha, tau.sq.beta, tau.sq.alpha, sigma.sq.mu, sigma.sq.p, kappa, and N. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the model, respectively. kappa is only relevant when family = 'NB'. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.

priors — a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, alpha.comm.normal, tau.sq.beta.ig, tau.sq.alpha.ig, sigma.sq.mu.ig, sigma.sq.p.ig, and kappa.unif. Community-level abundance (beta.comm) and detection (alpha.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances for the abundance coefficients are set to 100 and for the detection coefficients are set to 2.72. Community-level variance parameters for abundance (tau.sq.beta) and detection (tau.sq.alpha) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. sigma.sq.mu and sigma.sq.p are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts/slopes or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species

and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters.

`tuning` a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details.

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatial models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`...` currently no additional arguments

Value

An object of class `msNMix` that is a list comprised of:

`beta.comm.samples`
: a `coda` object of posterior samples for the community level abundance regression coefficients.

`alpha.comm.samples`
: a `coda` object of posterior samples for the community level detection regression coefficients.

`tau.sq.beta.samples`
: a `coda` object of posterior samples for the abundance community variance parameters.

`tau.sq.alpha.samples`
: a coda object of posterior samples for the detection community variance parameters.

`beta.samples`
: a coda object of posterior samples for the species level abundance regression coefficients.

`alpha.samples`
: a coda object of posterior samples for the species level detection regression coefficients.

`kappa.samples`
: a coda object of posterior samples for the species level abundance dispersion parameters. Only included when `family = 'NB'`.

`N.samples`
: a three-dimensional array of posterior samples for the latent abundance values for each species.

`mu.samples`
: a three-dimensional array of posterior samples for the latent expected abundance values for each species.

`sigma.sq.mu.samples`
: a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in `abund.formula`.

`sigma.sq.p.samples`
: a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples`
: a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation. Note that detection probability estimated values are not included in the model object, but can be extracted using `fitted()`.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Yamaura, Y., Royle, J. A., Shimada, N., Asanuma, S., Sato, T., Taki, H., & Makino, S. I. (2012). Biodiversity of man-made open habitats in an underused country: a class of multispecies abundance models for count data. Biodiversity and Conservation, 21(6), 1365-1380.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(5, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
# Abundance
beta.mean <- c(0, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Detection
alpha.mean <- c(0, 0.5, 0.8)
tau.sq.alpha <- c(0.2, 1, 1.5)
p.det <- length(alpha.mean)
# Random effects
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
                 mu.RE = mu.RE, p.RE = p.RE, sp = FALSE, family = 'Poisson')

y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.re <- dat$X.re
X.p.re <- dat$X.p.re

# Package all data into a list
abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')
det.covs <- list(det.cov.1 = as.data.frame(X.p[, , 2]),
                 det.cov.2 = as.data.frame(X.p[, , 3]))
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs)
prior.list <- list(beta.comm.normal = list(mean = rep(0, p.abund),
                                           var = rep(100, p.abund)),
```

```
                    alpha.comm.normal = list(mean = rep(0, p.det),
                                             var = rep(2.72, p.det)),
                    tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                    tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
inits.list <- list(beta.comm = 0, alpha.comm = 0,
                    beta = 0, alpha = 0,
                    tau.sq.beta = 0.5, tau.sq.alpha = 0.5,
                    N = apply(y, c(1, 2), max, na.rm = TRUE))
tuning.list <- list(beta = 0.5, alpha = 0.5)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- msNMix(abund.formula = ~ abund.cov.1,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              n.batch = n.batch,
              inits = inits.list,
              priors = prior.list,
              tuning = tuning.list,
              batch.length = batch.length,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 1,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)

summary(out, level = 'community')
```

---

neonDWP *Distance sampling data of 16 bird species observed in the Disney Wilderness Preserve in 2018 in Florida, USA*

Description

Distance sampling data of 16 bird species in 2018 in the Disney Wilderness Preserve in Florida, USA. These data were collected as part of the National Ecological Observatory Network (NEON). Data were collected at 90 sites where observers recorded the number of all bird species observed during a six minute, unlimited radius point count survey once during the breeding season. Distance of each individual bird to the observer was recorded using a laser rangefinder. For modeling, we binned the distance measurements into 4 distance bins, and only used observations within 250m. The 16 species included in the data set are as follows: (1) EATO (Eastern Towhee); (2) EAME (Eastern Meadowlark); (3) AMCR (American Crow); (4) BACS (Bachman's Sparrow); (5) CARW (Carolina Wren); (6) COGD (Common Ground Dove); (7) CONI (Common Nighthawk); (8) COYE (Common Yellowthroat); (9) EABL (Eastern Bluebird); (10) GCFL (Great-crested Flycatcher); (11) MODO (Mourning Dover); (12) NOCA (Northern Cardinal); (13) NOMO (Northern

Mockingbird); (14) RBWO (Red-bellied Woodpecker); (15) RHWO (Red-headed Woodpecker); (16) WEVI (White-eyed Vireo).

Usage

```
data(neonDWP)
```

Format

neonDWP is a list with five elements:

y: a three-dimensional array of distance sampling data with dimensions of species (16), sites (90) and distance bin (4).

covs: a data frame with 90 rows and four columns consisting of covariates for use in modeling abundance and/or detection.

dist.breaks: a vector of five values indicating the break points of the four distance bands in the data.

offset: an offset used to scale the 250m radius point count surveys to ha, such that resulting estimates will be the number of individuals per ha.

coords: a numeric matrix with 373 rows and two columns containing the site coordinates (Easting and Northing) in UTM Zone 17N. The EPSG is 32617.

Source

NEON (National Ecological Observatory Network). Breeding landbird point counts (DP1.10003.001), RELEASE-2023. https://doi.org/10.48443/00pg-vm19. Dataset accessed from https://data.neonscience.org on May 25, 2023

References

Barnett, D. T., Duffy, P. A., Schimel, D. S., Krauss, R. E., Irvine, K. M., Davis, F. W.,Gross, J. E., Azuaje, E. I., Thorpe, A. S., Gudex-Cross, D., et al. (2019). The terrestrial organism and biogeochemistry spatial sampling design for the national ecological observatory network. Ecosphere, 10(2):e02540.

---

neonPredData *Land cover covariates and coordinates at a 1ha resolution across Disney Wilderness Preserve*

Description

Land cover covariates (forest cover and grassland cover) extracted at a 1km resolution across the Disney Wildnerss Preserve for use in predicting density across the park. Land cover data come from USGS EROS.

Usage

```
data(neonPredData)
```

Format

neonPredData is a data frame with four columns:

forest: proportion of forest cover within 1km radius.

grass: proportion of grassland cover within 1km radius.

Easting: the x coordinate of the point. The EPSG is 32617 (UTM Zone 17N).

Northing: the y coordinate of the point. The EPSG is 32617 (UTM Zone 17N).

Source

USGS Earth Resources Observation and Science Ceneter https://www.usgs.gov/centers/eros

---

NMix *Function for Fitting Single-Species N-mixture Models*

Description

Function for fitting single-species N-mixture models.

Usage

```
NMix(abund.formula, det.formula, data, inits, priors, tuning,
     n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
     n.omp.threads = 1, verbose = TRUE,
     n.report = 100, n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
     n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, abund.covs, det.covs, and offset. y is a matrix or data frame of the observed count values, with first dimension equal to the number of sites (J) and second dimension equal to the maximum number of replicates at a given site. abund.covs is a matrix or data frame containing the variables used in the abundance portion of the model, with J rows for each column (variable). det.covs is a list of variables included in the detection portion of the model. Each list element is a different detection covariate, which can be site-level or observational-level.

Site-level covariates are specified as a vector of length J while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicates at a given site. `offset` is an offset to use in the abundance model (e.g., an area offset). This can be either a single value or a vector with an offset for each site (e.g., if survey area differed in size).

`inits` a list with each tag corresponding to a parameter name. Valid tags are `N`, `beta`, `alpha`, `kappa`, `sigma.sq.mu`, and `sigma.sq.p`. The value portion of each tag is the parameter's initial value. `sigma.sq.mu` and `sigma.sq.p` are only relevant when including random effects in the abundance and detection portion of the model, respectively. `kappa` is only relevant when `family = 'NB'`. See `priors` description for definition of each parameter name. Additionally, the tag `fix` can be set to `TRUE` to fix the starting values across all chains. If `fix` is not specified (the default), starting values are varied randomly across chains.

`priors` a list with each tag corresponding to a parameter name. Valid tags are `beta.normal`, `alpha.normal`, `kappa.unif`, `sigma.sq.mu.ig`, and `sigma.sq.p.ig`. Abundance (`beta`) and detection (`alpha`) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances set to 100 for the abundance coefficients and 2.72 for the detection coefficients. `kappa` is the negative binomial dispersion parameter and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a vector of length two with the first and second elements corresponding to the lower and upper bounds of the uniform distribution. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts/slopes or of length one if priors are the same for all random effect variances.

`tuning` a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), and `kappa`. See Roberts and Rosenthal (2009) for details.

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hypterthreaded cores. Note, n.omp.threads > 1 might not work on some systems. Currently only relevant for spatial models.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report Metropolis sampler acceptance and MCMC progress.
n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class NMix that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance dispersion parameter. Only included when family = 'NB'.
N.samples	a coda object of posterior samples for the latent abundance values
mu.samples	a coda object of posterior samples for the latent expected abundance values
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.
sigma.sq.p.samples	a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in det.formula.
beta.star.samples	a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in abund.formula.
alpha.star.samples	a coda object of posterior samples for the detection random effects. Only included if random effects are specified in det.formula.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation. Note that detection probability estimated values are not included in the model object, but can be extracted using fitted().

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115. doi:10.1111/j.0006341X.2004.00142.x.

Examples

```
set.seed(1010)
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0, -1.5)
p.abund <- length(beta)
alpha <- c(0.5, 1.2, -0.5)
p.det <- length(alpha)
mu.RE <- list()
p.RE <- list()
phi <- 3/.6
sigma.sq <- 2
kappa <- 0.3
sp <- FALSE
cov.model <- 'exponential'
dist <- 'NB'
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               kappa = kappa, mu.RE = mu.RE, p.RE = p.RE, sp = sp,
               phi = phi, sigma.sq = sigma.sq, cov.model = cov.model,
               family = 'NB')

y <- dat$y
X <- dat$X
X.p <- dat$X.p

abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')

det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])

data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs)
```

```
# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.abund),
                                      var = rep(100, p.abund)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)),
                   kappa.unif = c(0, 10))
# Starting values
inits.list <- list(alpha = 0,
                   beta = 0,
                   kappa = kappa,
                   N = apply(y, 1, max, na.rm = TRUE))

n.batch <- 4
batch.length <- 25
n.burn <- 50
n.thin <- 1
n.chains <- 1

out <- NMix(abund.formula = ~ abund.cov.1,
            det.formula = ~ det.cov.1 + det.cov.2,
            data = data.list,
            n.batch = n.batch,
            batch.length = batch.length,
            inits = inits.list,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 1,
            n.burn = n.burn,
            n.thin = n.thin,
            n.chains = n.chains)
```

---

ppcAbund *Function for performing posterior predictive checks*

Description

Function for performing posterior predictive checks on spAbundance model objects.

Usage

```
ppcAbund(object, fit.stat, group, type = 'marginal', ...)
```

Arguments

object	an object of class NMix, spNMix, msNMix, lfMsNMix, sfMsNMix, abund, spAbund, msAbund, lfMsAbund, sfMsAbund, DS, spDS, msDS, lfMsDS, sfMsDS.
fit.stat	a quoted keyword that specifies the fit statistic to use in the posterior predictive check. Supported fit statistics are "freeman-tukey" and "chi-squared".

group — a positive integer indicating the way to group the abundance data for the posterior predictive check. Value 0 will not group the data and use the raw counts, 1 will group values by row (site), and value 2 will group values by column (replicate).

type — a character string indicating whether fitted values should be generated conditional on the estimated latent abundance values (type = 'conditional') estimated during the model or based on the marginal expected abundance values (type = 'marginal'). This is only relevant for N-mixture models.

... — currently no additional arguments

Value

An object of class ppcAbund that is a list comprised of:

fit.y — a numeric vector of posterior samples for the fit statistic calculated on the observed data when object is of class NMix, spNMix, abund, spAbund, DS, or spDS. When object is of class msNMix, lfMsNMix, sfMsNMix, msAbund, lfMsAbund, sfMsAbund, msDS, lfMsDS, sfMsDS, this is a numeric matrix with rows corresponding to posterior samples and columns corresponding to species.

fit.y.rep — a numeric vector of posterior samples for the fit statistic calculated on a replicate data set generated from the model when object is of class NMix, spNMix, abund, spAbund, DS, spDS. When object is of class msNMix, lfMsNMix, sfMsNMix, msAbund, lfMsAbund, sfMsAbund, msDS, lfMsDS, sfMsDS, this is a numeric matrix with rows corresponding to posterior samples and columns corresponding to species.

fit.y.group.quants — a matrix consisting of posterior quantiles for the fit statistic using the observed data for each unique element the fit statistic is calculated for (i.e., observations when group = 0, sites when group = 1, replicates when group = 2) when object is of class NMix, spNMix, abund, spAbund, DS, or spDS. When object is of class msNMix, lfMsNMix, sfMsNMix, msAbund, lfMsAbund, sfMsAbund, msDS, lfMsDS, sfMsDS, this is a three-dimensional array with the additional dimension corresponding to species.

fit.y.rep.group.quants — a matrix consisting of posterior quantiles for the fit statistic using the model replicated data for each unique element the fit statistic is calculated for (i.e., observations when group = 0, sites when group = 1, replicates when group = 2) when object is of class NMix, spNMix, abund, spAbund, DS, spDS. When object is of class msNMix, sfMsNMix, msAbund, lfMsAbund, sfMsAbund, msDS, lfMsDS, sfMsDS, this is a three-dimensional array with the additional dimension corresponding to species.

The return object will include additional objects used for standard extractor functions.

Note

ppcAbund will return an error for Gaussian or zero-inflated Gaussian models. For Gaussian models, standard residual diagnostics can be used to assess model fit.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(1010)
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0, -1.5)
p.abund <- length(beta)
alpha <- c(0.5, 1.2, -0.5)
p.det <- length(alpha)
mu.RE <- list()
p.RE <- list()
phi <- 3/.6
sigma.sq <- 2
kappa <- 0.3
sp <- FALSE
cov.model <- 'exponential'
dist <- 'NB'
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               kappa = kappa, mu.RE = mu.RE, p.RE = p.RE, sp = sp,
               phi = phi, sigma.sq = sigma.sq, cov.model = cov.model,
               family = 'NB')

y <- dat$y
X <- dat$X
X.p <- dat$X.p

abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')

det.covs <- list(det.cov.1 = X.p[, , 2], det.cov.2 = X.p[, , 3])

data.list <- list(y = y, abund.covs = abund.covs,
                  det.covs = det.covs)

# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.abund),
                                      var = rep(100, p.abund)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)),
                   kappa.unif = c(0, 10))
# Starting values
inits.list <- list(alpha = 0, beta = 0, kappa = kappa,
                   N = apply(y, 1, max, na.rm = TRUE))


tuning <- 0.5
```

```
n.batch <- 4
batch.length <- 25
n.burn <- 50
n.thin <- 1
n.chains <- 1

out <- NMix(abund.formula = ~ abund.cov.1,
            det.formula = ~ det.cov.1 + det.cov.2,
            data = data.list,
            n.batch = n.batch,
            batch.length = batch.length,
            inits = inits.list,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 1,
            n.burn = n.burn,
            n.thin = n.thin,
            n.chains = n.chains)

# Posterior predictive check
ppc.out <- ppcAbund(out, fit.stat = 'chi-squared', group = 0)
summary(ppc.out)
```

`predict.abund` *Function for prediction at new locations for univariate GLMMs*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'abund'.

Usage

```
## S3 method for class 'abund'
predict(object, X.0, ignore.RE = FALSE, z.0.samples, ...)
```

Arguments

`object`	an object of class abund
`X.0`	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance portion of the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in `abund`. The covariates should be organized in the same order

as they were specified in the corresponding formula argument of abund. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of abund. See example below. If there is only one replicate per location, the design matrix can be a two-dimensional matrix instead of a three-dimensional array.

`ignore.RE` logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If `TRUE`, unstructured random effects will be included. If `FALSE`, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.

`z.0.samples` a matrix with rows corresponding to MCMC samples and columns corresponding to prediction locations containing the full posterior samples of the predicted binary portion of a zero-inflated Gaussian model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of the species at the location. When using `spOccupancy` to generate the first stage samples of the zero-inflated Gaussian model, this is the object contained in the `z.0.samples` object of the predition function for the spOccupancy object. Ignored for all model types other than zero-inflated Gaussian.

`...` currently no additional arguments

Value

A list object of class `predict.abund`. The list consists of:

`mu.0.samples` a three-dimensional object of posterior predictive samples for the expected abundance values with dimensions corresponding to posterior predictive sample, site, and replicate. When there is no replication, this will be a two-dimensional matrix. Note if an offset was used when fitting the model with abund, the abundance values are reported per unit of the offset.

`y.0.samples` a three-dimensional object of posterior predictive samples for the abundance values with dimensions corresponding to posterior predictive sample, site, and replicate. When there is no replication, this will be a two-dimensional matrix. These will be in the same units as `mu.0.samples`.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(1010)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(1, J, replace = TRUE)
beta <- c(0, -1.5, 0.3, -0.8)
p.abund <- length(beta)
mu.RE <- list()
kappa <- 0.5
sp <- FALSE
family <- 'NB'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, family = 'NB')

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Prediction covariates
X.0 <- dat$X[pred.indx, , ]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])

abund.covs <- list(int = X[, , 1],
                   abund.cov.1 = X[, , 2],
                   abund.cov.2 = X[, , 3],
                   abund.cov.3 = X[, , 4])

data.list <- list(y = y, covs = abund.covs)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 100),
                   kappa.unif = c(0.001, 10))
# Starting values
inits.list <- list(beta = 0, kappa = kappa)

n.batch <- 5
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- abund(formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
             data = data.list,
             n.batch = n.batch,
             batch.length = batch.length,
             inits = inits.list,
             priors = prior.list,
             accept.rate = 0.43,
             n.omp.threads = 1,
```

```
              verbose = TRUE,
              n.report = 1,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)

# Predict at new locations ------------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0)
mu.0.quants <- apply(out.pred$mu.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(dat$mu[pred.indx], mu.0.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(mu.0.quants), max(mu.0.quants)))
segments(dat$mu[pred.indx], mu.0.quants[1, ], dat$mu[pred.indx], mu.0.quants[3, ])
lines(dat$mu[pred.indx], dat$mu[pred.indx])
```

`predict.DS` *Function for prediction at new locations for single-species hierarchical distance sampling models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'DS'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'DS'
predict(object, X.0, ignore.RE = FALSE, type = 'abundance', ...)
```

Arguments

`object`	an object of class DS
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `DS`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `DS`. Column names of all variables must match the names of variables used when fitting the model (for the intercept, use `'(Intercept)'`).
`ignore.RE`	logical value that specifies whether or not to remove random abundance (or detection if `type = 'detection'`) effects from the subsequent predictions. If `TRUE`, random effects will be included. If `FALSE`, random effects will be set to 0 and predictions will only be generated from the fixed effects.

type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict latent abundance and expected abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
...	currently no additional arguments

Value

A list object of class predict.DS. When type = 'abundance', the list consists of:

mu.0.samples	a coda object of posterior predictive samples for the expected abundance values, or expected abundance per unit area (i.e., density) values when an offset was used when fitting the model with DS().
N.0.samples	a coda object of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.

When type = 'detection', the list consists of:

sigma.0.samples	a coda object of posterior predictive samples for sigma (the parameter controlling detection probability).

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random intercept corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(123)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Abundance coefficients
beta <- c(1.0, 0.2, 0.3, -0.2)
p.abund <- length(beta)
```

```
# Detection coefficients
alpha <- c(-1.0, -0.3)
p.det <- length(alpha)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
sp <- FALSE
family <- 'NB'
kappa <- 0.1
offset <- 1.8
transect <- 'point'

dat <- simDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
             beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
             mu.RE = mu.RE, p.RE = p.RE, sp = sp, family = family,
             offset = offset, transect = transect)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, ]
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  offset = offset)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 10),
                   alpha.normal = list(mean = 0,
                                       var = 10),
                   kappa.unif = c(0, 100))
# Starting values
inits.list <- list(alpha = 0,
                   beta = 0,
                   kappa = 1)
# Tuning values
tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.2)

out <- DS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
          det.formula = ~ det.cov.1,
```

```
            data = data.list,
            n.batch = 10,
            batch.length = 25,
            inits = inits.list,
            family = 'NB',
            det.func = 'halfnormal',
            transect = 'point',
            tuning = tuning,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 100,
            n.burn = 100,
            n.thin = 1,
            n.chains = 1)
summary(out)

# Predict at new locations ------------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0)
mu.0.quants <- apply(out.pred$mu.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(dat$mu[pred.indx], mu.0.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(mu.0.quants), max(mu.0.quants)))
segments(dat$mu[pred.indx], mu.0.quants[1, ], dat$mu[pred.indx], mu.0.quants[3, ])
lines(dat$mu[pred.indx], dat$mu[pred.indx])
```

`predict.lfMsAbund` *Function for prediction at new locations for latent factor multivariate GLMMs*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'lfMsAbund'.

Usage

```
## S3 method for class 'lfMsAbund'
predict(object, X.0, coords.0, ignore.RE = FALSE, z.0.samples, include.w = TRUE, ...)
```

Arguments

`object`	an object of class lfMsAbund
`X.0`	the design matrix of covariates at the prediction locations. This can be either a two-dimensional matrix with rows corresponding to sites and columns corresponding to covariates, or can be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in

	the model. If random effects are included in the the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in `lfMsAbund`. The covariates should be organized in the same order as they were specified in the corresponding formula argument of `lfMsAbund`. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of `lfMsAbund`. See example below.
`coords.0`	the spatial coordinates corresponding to `X.0`. Note that `spOccupancy` assumes coordinates are specified in a projected coordinate system.
`ignore.RE`	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If `TRUE`, unstructured random effects will be included. If `FALSE`, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.
`z.0.samples`	a three-dimensional array with dimensions corresponding to MCMC samples, species, and prediction locations. The array contains the full posterior samples of the predicted binary portion of a Gaussian zero-inflated model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of each species at the location. When using `spOccupancy` to generate the first stage samples of the zi-Gaussian model, this is the object contained in the `z.0.samples` object of the predition function for the spOccupancy object. Ignored for all model types other than zi-Gaussian.
`include.w`	a logical value used to indicate whether the latent random effects should be included in the predictions. By default, this is set to `TRUE`. If set to `FALSE`, predictions are given using the covariates and any unstructured random effects in the model. If `FALSE`, the `coords.0` argument is not required.
`...`	currently no additional arguments

Value

A list object of class `predict.lfMsAbund`. The list consists of:

`mu.0.samples`	a three or four-dimensional object of posterior predictive samples for the expected abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. Note if an offset was used when fitting the model with `lfMsAbund`, the abundance values are reported per unit of the offset.
`y.0.samples`	a three or four-dimensional object of posterior predictive samples for the abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. These will be in the same units as `mu.0.samples`.
`w.0.samples`	a three-dimensional array of posterior predictive samples for the latent factors.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels,

random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
mu.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- FALSE
kappa <- runif(n.sp, 0.1, 1)
factor.model <- TRUE
n.factors <- 3

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB',
                  factor.model = factor.model, n.factors = n.factors)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, , drop = FALSE]
# Occupancy covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Coordinates
coords <- dat$coords[-pred.indx, ]
# Prediction values
y.0 <- dat$y[, pred.indx, , drop = FALSE]
X.0 <- dat$X[pred.indx, , , drop = FALSE]
coords.0 <- dat$coords[pred.indx, ]

# Package all data into a list
covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2])
data.list <- list(y = y, covs = covs, coords = coords)
```

```
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   kappa = 0.5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1, lambda = 0.5, w = 0.5)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- lfMsAbund(formula = ~ abund.cov.1,
                 data = data.list,
                 n.batch = n.batch,
                 inits = inits.list,
                 priors = prior.list,
                 tuning = tuning.list,
                 batch.length = batch.length,
                 n.omp.threads = 1,
                 n.factors = 1,
                 verbose = TRUE,
                 n.report = 1,
                 n.burn = n.burn,
                 n.thin = n.thin,
                 n.chains = n.chains)

# Predict at new locations
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.lfMsDS` *Function for prediction at new locations for latent factor multi-species hierarchical distance sampling models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'lfMsDS'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'lfMsDS'
predict(object, X.0, coords.0, ignore.RE = FALSE,
        type = 'abundance', include.w = TRUE, ...)
```

Arguments

object	an object of class lfMsDS
X.0	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if type = 'detection') portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in lfMsDS. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of lfMsDS. Column names must match the names of the variables used to fit the model (for the intercept, use '(Intercept)').
coords.0	the spatial coordinates corresponding to X.0. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.
ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
include.w	a logical value used to indicate whether the latent random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords.0 argument is not required.
...	currently no additional arguments

Value

A list object of class predict.lfMsDS. When type = 'abundance', the list consists of:

mu.0.samples	a three-dimensional array of posterior predictive samples for the expected abundance values, or expected abundance values per unit area (i.e., density) values when an offset was used when fitting the model with lfMsDS().
N.0.samples	a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.

When type = 'detection', the list consists of:

sigma.0.samples	a three-dimensional array of posterior predictive samples for sigma (the parameter controlling detection probability).

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE
family <- 'NB'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
```

```
factor.model <- TRUE
n.factors <- 3

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
                 mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
                 sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
                 offset = offset, transect = transect, factor.model = factor.model,
                 n.factors = n.factors)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Occupancy covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , drop = FALSE]
X.p.0 <- dat$X.p[pred.indx, , drop = FALSE]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
                   alpha.comm.normal = list(mean = 0,
                                            var = 10),
                   kappa.unif = list(0, 100),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, kappa = 1)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8, lambda = 1, w = 1)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1
```

```
out <- lfMsDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
              det.formula = ~ det.cov.1,
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              family = 'Poisson',
              det.func = 'halfnormal',
              transect = transect,
              tuning = tuning,
              n.factors = n.factors,
              priors = prior.list,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 10,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
summary(out, level = 'community')

# Predict at new locations ---------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.lfMsNMix` *Function for prediction at new locations for latent factor multi-species N-mixture models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'lfMsNMix'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'lfMsNMix'
predict(object, X.0, coords.0, ignore.RE = FALSE,
        type = 'abundance', include.w = TRUE, ...)
```

Arguments

`object`	an object of class lfMsNMix
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should

be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `lfMsNMix`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `lfMsNMix`. Column names must match the names of the variables used to fit the model (for the intercept, use `'(Intercept)'`).

`coords.0` the spatial coordinates corresponding to `X.0`. Note that `spOccupancy` assumes coordinates are specified in a projected coordinate system. This is not a required argument, but can be used if some of the prediction sites are the same sites as those used to fit the data, in which case the latent factors estimated during model fitting can be used to improve the predictions. If predicting at different sites than those used to fit the data, this argument will not have any influence on model results.

`ignore.RE` a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.

`type` a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

`include.w` a logical value used to indicate whether the latent random effects should be included in the predictions. By default, this is set to `TRUE`. If set to `FALSE`, predictions are given using the covariates and any unstructured random effects in the model. If `FALSE`, the `coords.0` argument is not required.

`...` currently no additional arguments

Value

A list object of class `predict.lfMsNMix`. When `type = 'abundance'`, the list consists of:

`mu.0.samples` a three-dimensional array of posterior predictive samples for the expected abundance values. Note these will be per unit area if an offset was used when fitting the model with `lfMsNMix()`.

`N.0.samples` a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as `mu.0.samples`.

When `type = 'detection'`, the list consists of:

`p.0.samples` a three-dimensional array of posterior predictive samples for the detection probability values.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels,

random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(400)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep<- sample(2:4, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
# Abundance
beta.mean <- c(0.2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -0.1)
tau.sq.alpha <- c(0.2, 0.3, 1)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
family <- 'Poisson'
n.factors <- 3

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
                 sp = FALSE, family = 'Poisson', factor.model = TRUE,
                 n.factors = n.factors)
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Coordinates
coords <- dat$coords[-pred.indx, ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
```

```
mu.0 <- dat$psi[, pred.indx]
coords.0 <- dat$coords[pred.indx, ]
# Package all data into a list
abund.covs <- X[, 2, drop = FALSE]
colnames(abund.covs) <- c('abund.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)

# Occupancy initial values
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   alpha.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Initial values
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
                   beta = 0,
                   alpha = 0,
                   tau.sq.beta = 1,
                   tau.sq.alpha = 1,
                   N = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning values
tuning <- list(beta = 0.3, alpha = 0.3, lambda = 0.5, w = 0.5)
n.batch <- 4
batch.length <- 25
accept.rate <- 0.43

out <- lfMsNMix(abund.formula = ~ abund.cov,
                det.formula = ~ det.cov.1 + det.cov.2,
                data = data.list,
                inits = inits.list,
                family = 'Poisson',
                n.factors = n.factors,
                n.batch = n.batch,
                batch.length = batch.length,
                accept.rate = 0.43,
                tuning = tuning,
                priors = prior.list,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1)

summary(out, level = 'community')

# Predict at new locations ---------------------------------------------
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

predict.msAbund *Function for prediction at new locations for multivariate GLMMs*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'msAbund'.

Usage

```
## S3 method for class 'msAbund'
predict(object, X.0, ignore.RE = FALSE, z.0.samples, ...)
```

Arguments

object	an object of class msAbund
X.0	the design matrix of covariates at the prediction locations. This can be either a two-dimensional matrix with rows corresponding to sites and columns corresponding to covariates, or can be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in the model. If random effects are included in the the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in msAbund. The covariates should be organized in the same order as they were specified in the corresponding formula argument of msAbund. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of msAbund. See example below.
ignore.RE	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If TRUE, unstructured random effects will be included. If FALSE, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.
z.0.samples	a three-dimensional array with dimensions corresponding to MCMC samples, species, and prediction locations. The array contains the full posterior samples of the predicted binary portion of a Gaussian zero-inflated model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of each species at the location. When using spOccupancy to generate the first stage samples of the zi-Gaussian model, this is the object contained in the z.0.samples object of the predition function for the spOccupancy object. Ignored for all model types other than zi-Gaussian.
...	currently no additional arguments

Value

A list object of class `predict.msAbund`. The list consists of:

`mu.0.samples`	a three or four-dimensional object of posterior predictive samples for the expected abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. Note if an offset was used when fitting the model with `msAbund`, the abundance values are reported per unit of the offset.
`y.0.samples`	a three or four-dimensional object of posterior predictive samples for the abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. These will be in the same units as `mu.0.samples`.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
mu.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- FALSE
kappa <- runif(n.sp, 0.1, 1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB')
```

```
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, , drop = FALSE]
# Occupancy covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Prediction values
y.0 <- dat$y[, pred.indx, , drop = FALSE]
X.0 <- dat$X[pred.indx, , , drop = FALSE]

# Package all data into a list
covs <- list(int = X[, , 1], abund.cov.1 = X[, , 2])
data.list <- list(y = y, covs = covs)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   kappa = 0.5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- msAbund(formula = ~ abund.cov.1,
               data = data.list,
               n.batch = n.batch,
               inits = inits.list,
               priors = prior.list,
               tuning = tuning.list,
               batch.length = batch.length,
               n.omp.threads = 1,
               verbose = TRUE,
               n.report = 1,
               n.burn = n.burn,
               n.thin = n.thin,
               n.chains = n.chains)

# Predict at new locations
out.pred <- predict(out, X.0)
str(out.pred)
```

`predict.msDS` *Function for prediction at new locations for multi-species hierarchical distance sampling models*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'msDS'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'msDS'
predict(object, X.0, ignore.RE = FALSE, type = 'abundance', ...)
```

Arguments

object	an object of class msDS
X.0	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if type = 'detection') portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in msDS. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of msDS. Column names must match the names of the variables used to fit the model (for the intercept, use '(Intercept)').
ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
...	currently no additional arguments

Value

A list object of class predict.msDS. When type = 'abundance', the list consists of:

mu.0.samples	a three-dimensional array of posterior predictive samples for the expected abundance values, or expected abundance values per unit area (i.e., density) values when an offset was used when fitting the model with msDS().
N.0.samples	a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.

When type = 'detection', the list consists of:

sigma.0.samples	a three-dimensional array of posterior predictive samples for sigma (the parameter controlling detection probability).

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE
family <- 'Poisson'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
```

```
factor.model <- FALSE

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
                 mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
                 sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
                 offset = offset, transect = transect, factor.model = factor.model)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Occupancy covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , drop = FALSE]
X.p.0 <- dat$X.p[pred.indx, , drop = FALSE]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
   alpha.comm.normal = list(mean = 0,
               var = 10),
                   kappa.unif = list(0, 100),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
   alpha = 0, kappa = 1)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- msDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
            det.formula = ~ det.cov.1,
```

```
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              family = 'Poisson',
              det.func = 'halfnormal',
              transect = transect,
              tuning = tuning,
              priors = prior.list,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 10,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
summary(out, level = 'community')

# Predict at new locations ------------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0)
str(out.pred)
```

`predict.msNMix` *Function for prediction at new locations for multi-species N-mixture models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'msNMix'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'msNMix'
predict(object, X.0, ignore.RE = FALSE, type = 'abundance', ...)
```

Arguments

`object`	an object of class msNMix
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `msNMix`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `msNMix`. Column names must match the names of the variables used to fit the model (for the intercept, use `'(Intercept)'`).

ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.
...	currently no additional arguments
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

Value

A list object of class `predict.msNMix`. When `type = 'abundance'`, the list consists of:

mu.0.samples	a three-dimensional array of posterior predictive samples for the expected abundance values. Note these will be per unit area if an offset was used when fitting the model with `msNMix()`.
N.0.samples	a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as `mu.0.samples`.

When `type = 'detection'`, the list consists of:

p.0.samples	a three-dimensional array of posterior predictive samples for the detection probability values.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(400)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep<- sample(2:4, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
```

```
# Abundance
beta.mean <- c(0.2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -0.1)
tau.sq.alpha <- c(0.2, 0.3, 1)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
family <- 'Poisson'

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
                 sp = FALSE, family = 'Poisson')
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
mu.0 <- dat$psi[, pred.indx]
# Package all data into a list
abund.covs <- X[, 2, drop = FALSE]
colnames(abund.covs) <- c('abund.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs)

# Occupancy initial values
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   alpha.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Initial values
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
                   beta = 0,
                   alpha = 0,
                   tau.sq.beta = 1,
                   tau.sq.alpha = 1,
                   N = apply(y, c(1, 2), max, na.rm = TRUE))
```

```
# Tuning values
tuning <- list(beta = 0.3, alpha = 0.3)
n.batch <- 4
batch.length <- 25
accept.rate <- 0.43

out <- msNMix(abund.formula = ~ abund.cov,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              inits = inits.list,
              family = 'Poisson',
              n.batch = n.batch,
              batch.length = batch.length,
              accept.rate = 0.43,
              tuning = tuning,
              priors = prior.list,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 1)

summary(out, level = 'community')

# Predict at new locations ---------------------------------------------
out.pred <- predict(out, X.0)
str(out.pred)
```

`predict.NMix` *Function for prediction at new locations for single-species N-mixture models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'NMix'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'NMix'
predict(object, X.0, ignore.RE = FALSE, type = 'abundance', ...)
```

Arguments

`object`	an object of class NMix
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `NMix`. Columns should correspond to

	the order of how covariates were specified in the corresponding formula argument of NMix. Column names of all variables must match the names of variables used when fitting the model (for the intercept, use '(Intercept)').
ignore.RE	logical value that specifies whether or not to remove random abundance (or detection if type = 'detection') effects from the subsequent predictions. If TRUE, random effects will be included. If FALSE, random effects will be set to 0 and predictions will only be generated from the fixed effects.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict latent abundance and expected abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
...	currently no additional arguments

Value

A list object of class predict.NMix. When type = 'abundance', the list consists of:

mu.0.samples	a coda object of posterior predictive samples for the expected abundance values. Note these will be per unit area if an offset was used when fitting the model with NMix()
N.0.samples	a coda object of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.

When type = 'detection', the list consists of:

p.0.samples	a coda object of posterior predictive samples for the detection probability values.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random intercept corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
set.seed(100)
# Simulate Data ------------------------------------------------------------
J.x <- 10
J.y <- 10
```

```
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, 2)
p.abund <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               sp = FALSE)
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]

# Package all data into a list
abund.covs <- X[, 2, drop = FALSE]
colnames(abund.covs) <- c('abund.cov')
det.covs <- list(det.cov = X.p[, , 2])
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs)
# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.abund),
                                      var = rep(100, p.abund)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)),
                   kappa.unif = c(0.001, 10))
# Initial values
inits.list <- list(alpha = rep(0, p.det),
                   beta = rep(0, p.abund),
                   kappa = 0.5,
                   N = apply(y, 1, max, na.rm = TRUE))

n.batch <- 10
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- NMix(abund.formula = ~ abund.cov,
            det.formula = ~ det.cov,
            data = data.list,
            inits = inits.list,
            n.batch = n.batch,
            batch.length = batch.length,
            family = 'Poisson',
            priors = prior.list,
            n.omp.threads = 1,
            verbose = TRUE,
```

```
            n.report = 1,
            n.burn = n.burn,
            n.thin = n.thin,
            n.chains = n.chains)

summary(out)

# Predict at new locations ------------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov')
out.pred <- predict(out, X.0)
mu.0.quants <- apply(out.pred$mu.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(dat$mu[pred.indx], mu.0.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(mu.0.quants), max(mu.0.quants)))
segments(dat$mu[pred.indx], mu.0.quants[1, ], dat$mu[pred.indx], mu.0.quants[3, ])
lines(dat$mu[pred.indx], dat$mu[pred.indx])
```

`predict.sfMsAbund` *Function for prediction at new locations for spatial factor multivariate GLMMs*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'sfMsAbund'.

Usage

```
## S3 method for class 'sfMsAbund'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        z.0.samples, include.sp = TRUE, ...)
```

Arguments

`object`	an object of class sfMsAbund
`X.0`	the design matrix of covariates at the prediction locations. This can be either a two-dimensional matrix with rows corresponding to sites and columns corresponding to covariates, or can be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in the model. If random effects are included in the the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in `sfMsAbund`. The covariates should be organized in the same order as they were specified in the corresponding formula argument of `sfMsAbund`. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of `sfMsAbund`. See example below.

coords.0	the spatial coordinates corresponding to X.0. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.
ignore.RE	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If TRUE, unstructured random effects will be included. If FALSE, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.
z.0.samples	a three-dimensional array with dimensions corresponding to MCMC samples, species, and prediction locations. The array contains the full posterior samples of the predicted binary portion of a zero-inflated Gaussian model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of each species at the location. When using spOccupancy to generate the first stage samples of the zero-inflated Gaussian model, this is the object contained in the z.0.samples object of the predition function for the spOccupancy object. Ignored for all model types other than zero-inflated Gaussian.
include.sp	a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords.0 argument is not required.
...	currently no additional arguments

Value

A list object of class predict.sfMsAbund. The list consists of:

mu.0.samples	a three or four-dimensional object of posterior predictive samples for the expected abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. Note if an offset was used when fitting the model with sfMsAbund, the abundance values are reported per unit of the offset.
y.0.samples	a three or four-dimensional object of posterior predictive samples for the abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate. These will be in the same units as mu.0.samples.
w.0.samples	a three-dimensional array of posterior predictive samples for the latent factors. Array dimensions correspond to MCMC sample, latent factor, and site.

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
mu.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- TRUE
kappa <- runif(n.sp, 0.1, 1)
factor.model <- TRUE
n.factors <- 3
cov.model <- 'spherical'
phi <- runif(n.factors, 3 / 1, 3 / .1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB',
                  factor.model = factor.model, n.factors = n.factors,
                  phi = phi, cov.model = cov.model)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, , drop = FALSE]
# Occupancy covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Coordinates
coords <- dat$coords[-pred.indx, ]
# Prediction values
```

```
y.0 <- dat$y[, pred.indx, , drop = FALSE]
X.0 <- dat$X[pred.indx, , , drop = FALSE]
coords.0 <- dat$coords[pred.indx, ]

# Package all data into a list
covs <- list(int = X[, , 1], abund.cov.1 = X[, , 2])
data.list <- list(y = y, covs = covs, coords = coords)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   phi.unif = list(a = 3 / 1, b = 3 / .1),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   kappa = 0.5,
                   phi = 3 / .5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1, lambda = 0.5, w = 0.5,
                    phi = 1)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- sfMsAbund(formula = ~ abund.cov.1,
                 data = data.list,
                 n.batch = n.batch,
                 inits = inits.list,
                 priors = prior.list,
                 tuning = tuning.list,
                 batch.length = batch.length,
                 n.omp.threads = 1,
                 n.factors = 1,
                 verbose = TRUE,
                 n.neighbors = 5,
                 n.report = 1,
                 n.burn = n.burn,
                 n.thin = n.thin,
                 n.chains = n.chains)

# Predict at new locations
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.sfMsDS` *Function for prediction at new locations for spatial factor multi-species hierarchical distance sampling models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'sfMsDS'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'sfMsDS'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        type = 'abundance', include.sp = TRUE, ...)
```

Arguments

`object`	an object of class sfMsDS
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `sfMsDS`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `sfMsDS`. Column names must match the names of the variables used to fit the model (for the intercept, use `'(Intercept)'`).
`coords.0`	the spatial coordinates corresponding to `X.0`. Note that `spOccupancy` assumes coordinates are specified in a projected coordinate system.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report sampling progress.
`ignore.RE`	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.
`type`	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
`include.sp`	a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to `TRUE`. If set to `FALSE`, predictions are given using the covariates and any unstructured random effects in the model. If `FALSE`, the `coords` argument is not required.
`...`	currently no additional arguments

Value

A list object of class `predict.sfMsDS`. When `type = 'abundance'`, the list consists of:

`mu.0.samples` a three-dimensional array of posterior predictive samples for the expected abundance values, or expected abundance values per unit area (i.e., density) values when an offset was used when fitting the model with `sfMsDS()`.

`N.0.samples` a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as `mu.0.samples`.

`w.0.samples` a three-dimensional array of posterior predictive samples for the spatial latent factors.

When `type = 'detection'`, the list consists of:

`sigma.0.samples`
a three-dimensional array of posterior predictive samples for sigma (the parameter controlling detection probability).

The return object will include additional objects used for standard extractor functions.

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
```

```
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
family <- 'Poisson'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
factor.model <- TRUE
n.factors <- 3
phi <- runif(n.factors, 3 / 1, 3 / .2)
cov.model <- 'exponential'

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
                 mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
                 sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
                 offset = offset, transect = transect, factor.model = factor.model,
                 n.factors = n.factors)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Occupancy covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , drop = FALSE]
X.p.0 <- dat$X.p[pred.indx, , drop = FALSE]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
dist.breaks <- dat$dist.breaks


covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
```

```
                  covs = covs,
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
                   alpha.comm.normal = list(mean = 0, var = 10),
                   kappa.unif = list(0, 100),
                   phi.unif = list(3 / 1, 3 / .1),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, kappa = 1, phi = 3 / .5)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8, lambda = 1, w = 1, phi = 0.8)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- sfMsDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
              det.formula = ~ det.cov.1,
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              family = 'Poisson',
              det.func = 'halfnormal',
              transect = transect,
              tuning = tuning,
              cov.model = 'exponential',
              NNGP = TRUE,
              n.neighbors = 5,
              n.factors = n.factors,
              priors = prior.list,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 10,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
summary(out, level = 'community')

# Predict at new locations ---------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.sfMsNMix` *Function for prediction at new locations for spatial factor multi-species N-mixture models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'sfMsNMix'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'sfMsNMix'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100,
        ignore.RE = FALSE, type = 'abundance',
        include.sp = TRUE, ...)
```

Arguments

`object`	an object of class sfMsNMix
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `sfMsNMix`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `sfMsNMix`. Column names must match the names of the variables used to fit the model (for the intercept, use `'(Intercept)'`).
`coords.0`	the spatial coordinates corresponding to `X.0`. Note that `spOccupancy` assumes coordinates are specified in a projected coordinate system.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report sampling progress.
`ignore.RE`	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, random effects will be ignored and prediction will only use the fixed effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the random effect.

type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict expected abundance and latent abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
include.sp	a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords.0 argument is not required.
...	currently no additional arguments

Value

A list object of class predict.sfMsNMix. When type = 'abundance', the list consists of:

mu.0.samples	a three-dimensional array of posterior predictive samples for the expected abundance values. Note these will be per unit area if an offset was used when fitting the model with sfMsNMix().
N.0.samples	a three-dimensional array of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.
w.0.samples	a three-dimensional array of posterior predictive samples for the spatial latent factors.

When type = 'detection', the list consists of:

p.0.samples	a three-dimensional array of posterior predictive samples for the detection probability values.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

Examples

```
set.seed(400)
J.x <- 8
J.y <- 8
J <- J.x * J.y
```

```
n.rep<- sample(2:4, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
# Abundance
beta.mean <- c(0.2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -0.1)
tau.sq.alpha <- c(0.2, 0.3, 1)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
family <- 'Poisson'
n.factors <- 3
phi <- runif(n.factors, 3 / 1, 3 / .1)

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp,
                 beta = beta, alpha = alpha, sp = TRUE,
                 family = 'Poisson', factor.model = TRUE,
                 n.factors = n.factors, phi = phi, cov.model = 'exponential')
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Coordinates
coords <- dat$coords[-pred.indx, ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
mu.0 <- dat$psi[, pred.indx]
coords.0 <- dat$coords[pred.indx, ]
# Package all data into a list
abund.covs <- X[, 2, drop = FALSE]
colnames(abund.covs) <- c('abund.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)

# Initial values
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
```

```
                    alpha.comm.normal = list(mean = 0, var = 2.72),
                    tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                    tau.sq.alpha.ig = list(a = 0.1, b = 0.1),
                    phi.unif = list(a = 3 / 1, 3 / .1))
# Initial values
inits.list <- list(alpha.comm = 0,
                    beta.comm = 0,
                    beta = 0,
                    alpha = 0,
                    phi = 3 / .5,
                    tau.sq.beta = 1,
                    tau.sq.alpha = 1,
                    N = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning values
tuning <- list(beta = 0.3, alpha = 0.3, lambda = 0.5, w = 0.5, phi = 1.5)
n.batch <- 4
batch.length <- 25
accept.rate <- 0.43

out <- sfMsNMix(abund.formula = ~ abund.cov,
                det.formula = ~ det.cov.1 + det.cov.2,
                data = data.list,
                inits = inits.list,
                family = 'Poisson',
                n.factors = n.factors,
                n.batch = n.batch,
                batch.length = batch.length,
                accept.rate = 0.43,
                cov.model = 'exponential',
                n.neighbors = 5,
                tuning = tuning,
                priors = prior.list,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1)

summary(out, level = 'community')

# Predict at new locations ------------------------------------------------
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.spAbund` *Function for prediction at new locations for univariate spatial GLMMs*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'spAbund'.

Usage

```
## S3 method for class 'spAbund'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100,
        ignore.RE = FALSE, z.0.samples, include.sp = TRUE, ...)
```

Arguments

object	an object of class spAbund
X.0	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in the model. If random effects are included in the spAbundance portion of the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in spAbund. The covariates should be organized in the same order as they were specified in the corresponding formula argument of spAbund. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of spAbund. See example below. If there is only one replicate per location, the design matrix can be a two-dimensional matrix instead of a three-dimensional array.
coords.0	the spatial coordinates corresponding to X.0. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.
ignore.RE	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If TRUE, unstructured random effects will be included. If FALSE, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.
z.0.samples	a matrix with rows corresponding to MCMC samples and columns corresponding to prediction locations containing the full posterior samples of the predicted binary portion of a zero-inflated Gaussian model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of the species at the location. When using spOccupancy to generate the first stage samples of the zero-inflated Gaussian model, this is the object contained in the z.0.samples object of the predition function for the spOccupancy object. Ignored for all model types other than zero-inflated Gaussian.

include.sp — a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords.0 argument is not required.

... — currently no additional arguments

Value

A list object of class predict.spAbund. The list consists of:

mu.0.samples — a three-dimensional object of posterior predictive samples for the expected spAbundance values with dimensions corresponding to posterior predictive sample, site, and replicate. Note if an offset was used when fitting the model with spAbund, the abundance values are reported per unit of the offset.

y.0.samples — a three-dimensional object of posterior predictive samples for the spAbundance values with dimensions corresponding to posterior predictive sample, site, and replicate. These will be in the same units as mu.0.samples.

w.0.samples — a coda object of posterior predictive samples for the latent spatial random effects.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

Examples

```
set.seed(1010)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(1, J, replace = TRUE)
beta <- c(0, -1.5, 0.3, -0.8)
p.spAbund <- length(beta)
mu.RE <- list()
kappa <- 0.5
sp <- TRUE
sigma.sq <- 0.5
```

```
phi <- 3 / .5
family <- 'NB'
cov.model = 'exponential'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, family = 'NB',
                sigma.sq = sigma.sq, phi = phi, cov.model = cov.model)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Prediction covariates
X.0 <- dat$X[pred.indx, , ]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])

abund.covs <- list(int = X[, , 1],
                   abund.cov.1 = X[, , 2],
                   abund.cov.2 = X[, , 3],
                   abund.cov.3 = X[, , 4])

data.list <- list(y = y, covs = abund.covs, coords = coords)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 100),
                   kappa.unif = c(0.001, 10))
# Starting values
inits.list <- list(beta = 0, kappa = kappa)

n.batch <- 5
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- spAbund(formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
               data = data.list,
               n.batch = n.batch,
               batch.length = batch.length,
               inits = inits.list,
               priors = prior.list,
               accept.rate = 0.43,
               n.neighbors = 5,
               cov.model = cov.model,
               n.omp.threads = 1,
               verbose = TRUE,
               n.report = 1,
               n.burn = n.burn,
               n.thin = n.thin,
               n.chains = n.chains)

# Predict at new locations ---------------------------------------------
```

```
colnames(X.0) <- c('intercept', 'abund.cov', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0, coords.0)
mu.0.quants <- apply(out.pred$mu.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(dat$mu[pred.indx], mu.0.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(mu.0.quants), max(mu.0.quants)))
segments(dat$mu[pred.indx], mu.0.quants[1, ], dat$mu[pred.indx], mu.0.quants[3, ])
lines(dat$mu[pred.indx], dat$mu[pred.indx])
```

`predict.spDS` *Function for prediction at new locations for single-species spatially-explicit hierarchical distance sampling models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'spDS'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'spDS'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        type = 'abundance', include.sp = TRUE, ...)
```

Arguments

`object`	an object of class spDS
`X.0`	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if `type = 'detection'`) portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in `spDS`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `spDS`. Column names of all variables must match the names of variables used when fitting the model (for the intercept, use `'(Intercept)'`).
`coords.0`	the spatial coordinates corresponding to `X.0`. Note that `spAbundance` assumes coordinates are specified in a projected coordinate system.
`n.omp.threads`	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.
`verbose`	if `TRUE`, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
`n.report`	the interval to report sampling progress.

ignore.RE — logical value that specifies whether or not to remove random abundance (or detection if type = 'detection') effects from the subsequent predictions. If TRUE, random effects will be included. If FALSE, random effects will be set to 0 and predictions will only be generated from the fixed effects.

type — a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict latent abundance and expected abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

include.sp — a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords argument is not required.

... — currently no additional arguments

Value

A list object of class predict.spDS. When type = 'abundance', the list consists of:

mu.0.samples — a coda object of posterior predictive samples for the expected abundance values, or expected abundance per unit area (i.e., density) values when an offset was used when fitting the model with spDS().

N.0.samples — a coda object of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples

w.0.samples — a coda object of posterior predictive samples for the latent spatial random effects.

When type = 'detection', the list consists of:

sigma.0.samples — a coda object of posterior predictive samples for sigma (the parameter controlling detection probability).

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random intercept corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

Examples

```
set.seed(123)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Abundance coefficients
beta <- c(1.0, 0.2, 0.3, -0.2)
p.abund <- length(beta)
# Detection coefficients
alpha <- c(-1.0, -0.3)
p.det <- length(alpha)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
sp <- TRUE
phi <- 3 / .5
sigma.sq <- 0.8
cov.model <- 'exponential'
family <- 'NB'
kappa <- 0.1
offset <- 1.8
transect <- 'point'

dat <- simDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
             beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
             mu.RE = mu.RE, p.RE = p.RE, sp = sp, family = family,
             offset = offset, transect = transect, phi = phi, sigma.sq = sigma.sq,
             cov.model = cov.model)
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, ]
dist.breaks <- dat$dist.breaks
coords <- dat$coords[-pred.indx, ]
coords.0 <- dat$coords[pred.indx, ]

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
```

```
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 10),
                   alpha.normal = list(mean = 0,
                                       var = 10),
                   kappa.unif = c(0, 100),
                   phi.unif = c(3 / 1, 3 / .1),
                   sigma.sq.ig = c(2, 1))
# Starting values
inits.list <- list(alpha = 0,
                   beta = 0,
                   kappa = 1,
                   phi = 3 / .5,
                   sigma.sq = 1)
# Tuning values
tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.2, phi = 1, w = 1)

out <- spDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
            det.formula = ~ det.cov.1,
            data = data.list,
            n.batch = 10,
            batch.length = 25,
            inits = inits.list,
            family = 'NB',
            det.func = 'halfnormal',
            transect = 'point',
            cov.model = 'exponential',
            NNGP = TRUE,
            n.neighbors = 5,
            tuning = tuning,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 100,
            n.burn = 100,
            n.thin = 1,
            n.chains = 1)
summary(out)

# Predict at new locations ---------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3')
out.pred <- predict(out, X.0, coords.0)
mu.0.quants <- apply(out.pred$mu.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(dat$mu[pred.indx], mu.0.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(mu.0.quants), max(mu.0.quants)))
segments(dat$mu[pred.indx], mu.0.quants[1, ], dat$mu[pred.indx], mu.0.quants[3, ])
lines(dat$mu[pred.indx], dat$mu[pred.indx])
```

predict.spNMix *Function for prediction at new locations for single-species spatial N-mixture models*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'spNMix'. Prediction is possible for both the latent abundance state as well as detection.

Usage

```
## S3 method for class 'spNMix'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        type = 'abundance', include.sp = TRUE, ...)
```

Arguments

object	an object of class spNMix
X.0	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the abundance (or detection if type = 'detection') portion of the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in spNMix. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of spNMix. Column names of all variables must match the names of variables used when fitting the model (for the intercept, use '(Intercept)').
coords.0	the spatial coordinates corresponding to X.0. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.
ignore.RE	logical value that specifies whether or not to remove random abundance (or detection if type = 'detection') effects from the subsequent predictions. If TRUE, random effects will be included. If FALSE, random effects will be set to 0 and predictions will only be generated from the fixed effects.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'abundance' to predict latent abundance and expected abundance values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

include.sp — a logical value used to indicate whether spatial random effects should be included in the predictions. By default, this is set to TRUE. If set to FALSE, predictions are given using the covariates and any unstructured random effects in the model. If FALSE, the coords argument is not required.

... — currently no additional arguments

Value

A list object of class predict.spNMix. When type = 'abundance', the list consists of:

mu.0.samples — a coda object of posterior predictive samples for the expected abundance values. Note these will be per unit area if an offset was used when fitting the model with NMix()

N.0.samples — a coda object of posterior predictive samples for the latent abundance values. These will be in the same units as mu.0.samples.

w.0.samples — a coda object of posterior predictive samples for the latent spatial random effects.

When type = 'detection', the list consists of:

p.0.samples — a coda object of posterior predictive samples for the detection probability values.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random intercept corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

Examples

```
set.seed(200)
# Simulate Data ------------------------------------------------------------
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0.5, 1.5)
p.abund <- length(beta)
alpha <- c(0.5, 1.2, -0.5)
```

```
p.det <- length(alpha)
mu.RE <- list()
p.RE <- list()
phi <- runif(1, 3 / 1, 3 / .1)
sigma.sq <- runif(1, 0.2, 1.5)
kappa <- 0.5
sp <- TRUE
cov.model <- 'exponential'
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               kappa = kappa, mu.RE = mu.RE, p.RE = p.RE, sp = sp,
               phi = phi, sigma.sq = sigma.sq, cov.model = cov.model,
               family = 'NB')

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .5), replace = FALSE)
y <- dat$y[-pred.indx, ]
# Abundance covariates
X <- dat$X[-pred.indx, ]
# Prediction covariates
X.0 <- dat$X[pred.indx, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
mu.0 <- dat$mu[pred.indx]
w.0 <- dat$w[pred.indx]

abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')

det.covs <- list(det.cov.1 = X.p[, , 2], det.cov.2 = X.p[, , 3])

data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)

# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.abund),
                                      var = rep(100, p.abund)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)),
                   kappa.unif = c(0, 10))
# Starting values
inits.list <- list(alpha = alpha,
                   beta = beta,
                   kappa = kappa,
                   phi = 3 / 0.5,
                   sigma.sq = 1,
                   N = apply(y, 1, max, na.rm = TRUE))
# Tuning values
tuning.list <- list(phi = 0.5, kappa = 0.5, beta = 0.1, alpha = 0.1, w = 0.1)
```

```
n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- spNMix(abund.formula = ~ abund.cov.1,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              priors = prior.list,
              NNGP = TRUE,
              cov.model = 'spherical',
              n.neighbors = 10,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 1,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
summary(out)

# Predict at new locations ------------------------------------------------
colnames(X.0) <- c('intercept', 'abund.cov')
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

`predict.svcAbund` *Function for prediction at new locations for univariate Gaussian spatially-varying coefficient models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'svcAbund'.

Usage

```
## S3 method for class 'svcAbund'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        z.0.samples, ...)
```

Arguments

object	an object of class svcAbund
X.0	the design matrix of covariates at the prediction locations. This should include a column of 1s for the intercept if an intercept is included in the model. If random effects are included in the model, the levels of the random effects at the new locations should be included as a column in the design matrix. The ordering of the levels should match the ordering used to fit the data in svcAbund. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of svcAbund. Column names of all variables must match the names of variables used when fitting the model (for the intercept, use '(Intercept)').
coords.0	the spatial coordinates corresponding to X.0. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.
ignore.RE	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If TRUE, random effects will be included. If FALSE, random effects will be set to 0 and predictions will only be generated from the fixed effects.
z.0.samples	a matrix with rows corresponding to MCMC samples and columns corresponding to prediction locations containing the full posterior samples of the predicted binary portion of a zero-inflated Gaussian model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of the species at the location. When using spOccupancy to generate the first stage samples of the zero-inflated Gaussian model, this is the object contained in the z.0.samples object of the predition function for th spOccupancy object. Ignored for all model types other than zero-inflated Gaussian.
...	currently no additional arguments

Value

A list object of class predict.svcAbund. When type = 'abundance', the list consists of:

mu.0.samples	a coda object of posterior predictive samples for the expected abundance values.
y.0.samples	a coda object of posterior predictive samples for the abundance values.
w.0.samples	a three-dimensional array of posterior predictive samples for the spatially-varying coefficients, with dimensions corresponding to MCMC iteration, coefficient, and site.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random intercept corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>,

Examples

```
set.seed(1000)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Occurrence --------------------------
beta <- c(10, 0.5, -0.2, 0.75)
p <- length(beta)
mu.RE <- list()
# Spatial parameters ------------------
sp <- TRUE
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.4, 4)
phi <- runif(p.svc, 3/1, 3/0.7)
tau.sq <- 2

# Get all the data
dat <- simAbund(J.x = J.x, J.y = J.y, beta = beta, tau.sq = tau.sq,
                mu.RE = mu.RE, sp = sp, svc.cols = svc.cols, family = 'Gaussian',
                cov.model = cov.model, sigma.sq = sigma.sq, phi = phi)

# Prep the data for spAbundance -------------------------------------------
y <- dat$y
X <- dat$X
coords <- dat$coords

# Subset data for prediction if desired
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y.0 <- y[pred.indx, drop = FALSE]
X.0 <- X[pred.indx, , drop = FALSE]
coords.0 <- coords[pred.indx, ]
y <- y[-pred.indx, drop = FALSE]
X <- X[-pred.indx, , drop = FALSE]
```

```
coords <- coords[-pred.indx, ]

# Package all data into a list
covs <- cbind(X)
colnames(covs) <- c('int', 'cov.1', 'cov.2', 'cov.3')

# Data list bundle
data.list <- list(y = y, covs = covs, coords = coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 1000),
                   sigma.sq.ig = list(a = 2, b = 1), tau.sq = c(2, 1),
                   sigma.sq.mu.ig = list(a = 2, b = 1),
                   phi.unif = list(a = 3 / 1, b = 3 / 0.1))

# Starting values
inits.list <- list(beta = 0, alpha = 0,
                   sigma.sq = 1, phi = phi, tau.sq = 2, sigma.sq.mu = 0.5)
# Tuning
tuning.list <- list(phi = 1)

n.batch <- 10
batch.length <- 25
n.burn <- 100
n.thin <- 1
n.chains <- 3

out <- svcAbund(formula = ~ cov.1 + cov.2 + cov.3,
                svc.cols = svc.cols,
                data = data.list,
                n.batch = n.batch,
                batch.length = batch.length,
                inits = inits.list,
                priors = prior.list,
                accept.rate = 0.43,
                family = 'Gaussian',
                cov.model = "exponential",
                tuning = tuning.list,
                n.omp.threads = 1,
                verbose = TRUE,
                NNGP = TRUE,
                n.neighbors = 5,
                n.report = 25,
                n.burn = n.burn,
                n.thin = n.thin,
                n.chains = n.chains)

# Predict at new values ------------------------------------------------
out.pred <- predict(out, X.0, coords.0)

mu.0.means <- apply(out.pred$mu.0.samples, 2, mean)
mu.0 <- dat$mu[pred.indx]
plot(mu.0, mu.0.means, pch = 19)
abline(0, 1)
```

predict.svcMsAbund *Function for prediction at new locations for multivariate spatially-varying coefficient GLMMs*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'svcMsAbund'.

Usage

```
## S3 method for class 'svcMsAbund'
predict(object, X.0, coords.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE,
        z.0.samples, ...)
```

Arguments

object	an object of class svcMsAbund
X.0	the design matrix of covariates at the prediction locations. This can be either a two-dimensional matrix with rows corresponding to sites and columns corresponding to covariates, or can be a three-dimensional array, with dimensions corresponding to site, replicate, and covariate, respectively. Note that the first covariate should consist of all 1s for the intercept if an intercept is included in the model. If random effects are included in the the model, the levels of the random effects at the new locations/time periods should be included as an element of the three-dimensional array. The ordering of the levels should match the ordering used to fit the data in svcMsAbund. The covariates should be organized in the same order as they were specified in the corresponding formula argument of svcMsAbund. Names of the third dimension (covariates) of any random effects in X.0 must match the name of the random effects used to fit the model, if specified in the corresponding formula argument of svcMsAbund. See example below.
coords.0	the spatial coordinates corresponding to X.0. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
n.report	the interval to report sampling progress.

ignore.RE	logical value that specifies whether or not to remove unstructured random effects from the subsequent predictions. If TRUE, unstructured random effects will be included. If FALSE, unstructured random effects will be set to 0 and predictions will only be generated from the fixed effects.
z.0.samples	a three-dimensional array with dimensions corresponding to MCMC samples, species, and prediction locations. The array contains the full posterior samples of the predicted binary portion of a zero-inflated Gaussian model. In the context of abundance models, this typically corresponds to estimates of the presence or absence of each species at the location. When using spOccupancy to generate the first stage samples of the zero-inflated Gaussian model, this is the object contained in the z.0.samples object of the predition function for the spOccupancy object. Ignored for all model types other than zero-inflated Gaussian.
...	currently no additional arguments

Value

A list object of class predict.svcMsAbund. The list consists of:

mu.0.samples	a three or four-dimensional object of posterior predictive samples for the expected abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate.
y.0.samples	a three or four-dimensional object of posterior predictive samples for the abundance values with dimensions corresponding to posterior predictive sample, species, site, and replicate.
w.0.samples	a four-dimensional array of posterior predictive samples for the spatial factors for each spatially-varying coefficient. Dimensions correspond to MCMC sample, spatial factor, site, and spatially varying coefficient.

The return object will include additional objects used for standard extractor functions.

Note

When ignore.RE = FALSE, both sampled levels and non-sampled levels of random effects are supported for prediction. For sampled levels, the posterior distribution for the random effect corresponding to that level of the random effect will be used in the prediction. For non-sampled levels, random values are drawn from a normal distribution using the posterior samples of the random effect variance, which results in fully propagated uncertainty in predictions with models that incorporate random effects.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
```

```
J <- J.x * J.y
n.rep <- rep(1, J)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
mu.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- TRUE
factor.model <- TRUE
n.factors <- 2
svc.cols <- c(1, 2)
cov.model <- 'spherical'
tau.sq <- runif(n.sp, 0.1, 2)
phi <- runif(n.factors * length(svc.cols), 3 / 1, 3 / .1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, family = 'Gaussian', tau.sq = tau.sq,
                  factor.model = factor.model, n.factors = n.factors,
                  phi = phi, cov.model = cov.model, svc.cols = svc.cols)

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[, -pred.indx, drop = FALSE]
# Occupancy covariates
X <- dat$X[-pred.indx, , drop = FALSE]
# Coordinates
coords <- dat$coords[-pred.indx, ]
# Prediction values
y.0 <- dat$y[, pred.indx, drop = FALSE]
X.0 <- dat$X[pred.indx, , drop = FALSE]
coords.0 <- dat$coords[pred.indx, ]

# Package all data into a list
covs <- data.frame(abund.cov.1 = X[, 2])
data.list <- list(y = y, covs = covs, coords = coords)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   tau.sq.ig = list(a = .01, b = .01),
                   phi.unif = list(a = 3 / 1, b = 3 / .1),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   kappa = 0.5,
                   tau.sq = 1,
                   phi = 3 / .5,
                   tau.sq.beta = 1)
tuning.list <- list(kappa = 0.3, beta = 0.1, lambda = 0.5, w = 0.5,
                    phi = 1)
```

```
# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- svcMsAbund(formula = ~ abund.cov.1,
                  data = data.list,
                  n.batch = n.batch,
                  inits = inits.list,
                  priors = prior.list,
                  tuning = tuning.list,
                  batch.length = batch.length,
                  n.omp.threads = 1,
                  svc.cols = c(1, 2),
                  n.factors = n.factors,
                  cov.model = 'exponential',
                  family = 'Gaussian',
                  verbose = TRUE,
                  n.neighbors = 5,
                  n.report = 1,
                  n.burn = n.burn,
                  n.thin = n.thin,
                  n.chains = n.chains)

# Predict at new locations
out.pred <- predict(out, X.0, coords.0)
str(out.pred)
```

---

sfMsAbund *Function for Fitting Spatial Factor Multivariate Abundance GLMMs*

Description

The function sfMsAbund fits multivariate spatial abundance GLMMs with species correlations (i.e., a spatially-explicit abundace-based joint species distribution model). We use a spatial factor modeling approach. Currently, models are implemented using a Nearest Neighbor Gaussian Process. Future development may allow for running the models using full Gaussian Processes.

Usage

```
sfMsAbund(formula, data, inits, priors,
          tuning, cov.model = 'exponential', NNGP = TRUE,
          n.neighbors = 15, search.type = 'cb', n.factors,
          n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
          n.omp.threads = 1, verbose = TRUE, n.report = 100,
         n.burn = round(.10 * n.batch * batch.length), n.thin = 1, n.chains = 1,
          save.fitted = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, z, coords, and offset. y is a two or three-dimensional array of observed count data. The first dimension of the array is equal to the number of species and the second dimension is equal to the number of sites. If specified as a three-dimensional array, the third dimension corresponds to replicate observations at each site (e.g., sub-samples, repeated sampling over multiple seasons). covs is a list containing the variables used in the model. If a data frame, each row of covs is a site and each column is a variable. If a list, each list element is a different covariate, which can be site-level or observation-level. Site-level covariates are specified as a vector of length J, while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicate observations at a given site. coords is a $J \times 2$ matrix of the observation coordinates. Note that spAbundance assumes coordinates are specified in a projected coordinate system. For zero-inflated Gaussian models, the tag z is used to specify the binary component of the model and should have the same dimensions as y. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value, a vector with an offset for each site (e.g., if survey area differed in size), or a site x replicate matrix if more than one count is available at a given site.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta.comm, beta, tau.sq.beta, sigma.sq.mu, kappa, phi, lambda, nu, and tau.sq. nu is only specified if cov.model = "matern", kappa is only specified if family = 'NB', tau.sq is only specified for Gaussian and zero-inflated Gaussian models, and sigma.sq.mu is only specified if random effects are included in formula. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, tau.sq.beta.ig, sigma.sq.mu, kappa.unif, phi.unif, nu.unif, and tau.sq.ig. Community-level (beta.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances to 100. Community-level variance parameters (tau.sq.beta) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. The spatial

factor model fits n.factors independent spatial processes. The spatial decay phi and smoothness nu parameters for each latent factor are assumed to follow Uniform distributions. The hyperparameters of the Uniform are passed as a list with two elements, with both elements being vectors of length n.factors corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factors. The priors for the factor loadings matrix lambda are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen and Amemlya 2002). The upper triangular elements of the n.sp x n.factors matrix are fixed at 0 and the diagonal elements are fixed at 1. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1). sigma.sq.mu are the random effect variances random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters. tau.sq is the species-specific residual variance for Gaussian (or zero-inflated Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a list of length two, with the first and second element corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of species or a single value if priors are the same for all species.

tuning — a single numeric value representing the initial variance of the adaptive sampler for beta, alpha, beta.star (the abundance random effect values), kappa, phi, lambda. See Roberts and Rosenthal (2009) for details. Note that only phi and nu are tuned for Gaussian or zero-inflated Gaussian models.

cov.model — a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".

NNGP — if TRUE, model is fit with an NNGP. If FALSE, a full Gaussian process is used. See Datta et al. (2016) and Finley et al. (2019) for more information. For spatial factor models, only NNGP = TRUE is currently supported.

n.neighbors — number of neighbors used in the NNGP. Only used if NNGP = TRUE. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

search.type — a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: "cb" and "brute". The "cb" should generally be much faster. If locations do not have identical coordinate values on the

axis used for the nearest neighbor ordering then "cb" and "brute" should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then "cb" and "brute" might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.factors` the number of factors to use in the spatial factor model approach. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and the number of species in the modeled community.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for adaptive MCMC. Defaul is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the abundance. Currently supports `'NB'` (negative binomial), `'Poisson'` (Poisson), `'Gaussian'` (Gaussian), and `'zi-Gaussian'` (zero-inflated Gaussian).

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report Metropolis sampler acceptance and MCMC progress. Note this is specified in terms of batches and not overall samples for spatial models.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

`n.thin` the thinning interval for collection of MCMC samples. The thinning occurs after the `n.burn` samples are discarded. Default value is set to 1.

`n.chains` the number of chains to run in sequence.

`save.fitted` logical value indicating whether or not fitted values and likelihood values should be saved in the resulting model object. If `save.fitted = FALSE`, the components `y.rep.samples`, `mu.samples`, and `like.samples` will not be included in the model object, and subsequent functions for calculating WAIC, fitted values, and posterior predictive checks will not work, although they all can be calculated manually if desired. Setting `save.fitted = FALSE` can be useful when working with very large data sets to minimize the amount of RAM needed when fitting and storing the model object in memory.

`...` currently no additional arguments

Value

An object of class `sfMsAbund` that is a list comprised of:

`beta.comm.samples`
: a coda object of posterior samples for the community level regression coefficients.

`tau.sq.beta.samples`
: a coda object of posterior samples for the abundance community variance parameters.

`beta.samples`
: a coda object of posterior samples for the species level abundance regression coefficients.

`kappa.samples`
: a coda object of posterior samples for the species level abundance dispersion parameters. Only included when `family = 'NB'`.

`tau.sq.samples`
: a coda object of posterior samples for the Gaussian residual variance parameter. Only included when `family = 'Gaussian'` or `family = 'zi-Gaussian'`.

`theta.samples`
: a coda object of posterior samples for the spatial correlation parameters.

`lambda.samples`
: a coda object of posterior samples for the latent spatial factor loadings.

`y.rep.samples`
: a three or four-dimensional array of posterior samples for the fitted (replicate) values for each species with dimensions corresponding to MCMC sample, species, site, and replicate.

`mu.samples`
: a three or four-dimensional array of posterior samples for the expected abundance values for each species with dimensions corresponding to MCMC samples, species, site, and replicate.

`w.samples`
: a three-dimensional array of posterior samples for the latent effects for each latent factor. Array dimensions correspond to MCMC sample, latent factor, then site.

`sigma.sq.mu.samples`
: a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in `abund.formula`.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`like.samples`
: a three-dimensional array of posterior samples for the likelihood value associated with each site and species. Used for calculating WAIC.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Christensen, W. F., and Amemiya, Y. (2002). Latent variable analysis of multivariate spatial data. *Journal of the American Statistical Association*, 97(457), 302-317.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Random effects (two random intercepts)
mu.RE <- list(levels = c(10, 15),
              sigma.sq.mu = c(0.43, 0.5))
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- TRUE
n.factors <- 2
factor.model <- TRUE
phi <- runif(n.factors, 3/1, 3 / .1)
kappa <- runif(n.sp, 0.1, 1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB',
                  factor.model = factor.model, phi = phi,
                  cov.model = 'exponential', n.factors = n.factors)

y <- dat$y
X <- dat$X
X.re <- dat$X.re
coords <- dat$coords
```

```
# Package all data into a list
covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2],
             abund.factor.1 = X.re[, , 1],
             abund.factor.2 = X.re[, , 2])
data.list <- list(y = y, covs = covs, coords = coords)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   kappa.unif = list(a = 0, b = 10),
                   phi.unif = list(a = 3 / 1, b = 3 / .1),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0, beta = 0, kappa = 0.5,
                   tau.sq.beta = 1, phi = 3 / 0.5)

# Small
n.batch <- 2
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- sfMsAbund(formula = ~ abund.cov.1 + (1 | abund.factor.1) +
                             (1 | abund.factor.2),
                 data = data.list,
                 n.batch = n.batch,
                 inits = inits.list,
                 priors = prior.list,
                 NNGP = TRUE,
                 cov.model = 'exponential',
                 n.neighbors = 5,
                 n.factors = n.factors,
                 batch.length = batch.length,
                 n.omp.threads = 3,
                 verbose = TRUE,
                 n.report = 1,
                 n.burn = n.burn,
                 n.thin = n.thin,
                 n.chains = n.chains)
summary(out)
```

`sfMsDS` *Function for Fitting Spatial Factor Multi-Species Hierarchical Distance Sampling Models*

Description

Function for fitting spatial factor multi-species hierarchical distance sampling models.

Usage

```
sfMsDS(abund.formula, det.formula, data, inits, priors,
       tuning, cov.model = 'exponential', NNGP = TRUE,
       n.neighbors = 15, search.type = 'cb', n.factors,
       n.batch, batch.length, accept.rate = 0.43,
       family = 'Poisson', transect = 'line', det.func = 'halfnormal',
       n.omp.threads = 1, verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
       n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, dist.breaks, and offset. y is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. covs is a matrix or data frame containing the variables used in the abundance and/or the detection portion of the model, with J rows for each column (variable). dist.breaks is a vector of distances that denote the breakpoints of the distance bands. dist.breaks should have length equal to the third dimension of y plus one. offset is an offset that can be used to scale estimates from abundance per transect to density per some desired unit of measure. This can be either a single value or a vector with an offset value for each site (e.g., if transects differ in length). coords is a matrix or data frame with two columns that contain the spatial coordinates of each site. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
inits	a list with each tag corresponding to a parameter name. Valid tags are alpha.comm, beta.comm, beta, alpha, tau.sq.beta, tau.sq.alpha, sigma.sq.mu, sigma.sq.p, phi, nu, lambda, w, kappa, and N. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the model, respectively. kappa is only relevant when family = 'NB'. nu is only relevant if cov.model = "matern". The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, alpha.comm.normal, tau.sq.beta.ig, tau.sq.alpha.ig, sigma.sq.mu.ig,

sigma.sq.p.ig, kappa.unif, phi.unif, and nu.unif. Community-level abundance (beta.comm) and detection (alpha.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances are set to 100. Community-level variance parameters for abundance (tau.sq.beta) and detection (tau.sq.alpha) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. sigma.sq.mu and sigma.sq.p are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters. The spatial factor model fits n.factors independent spatial processes. The spatial decay phi and smoothness nu parameters for each latent factor are assumed to follow Uniform distributions. The hyperparameters of the Uniform are passed as a list with two elements, with both elements being vectors of length n.factors corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factors. The priors for the factor loadings matrix lambda are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen and Amemlya 2002). The upper triangular elements of the n.sp x n.factors matrix are fixed at 0 and the diagonal elements are fixed at 1. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1).

tuning — a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are beta, alpha, lambda (the latent factor loadings), w (the latent factors), beta.star (the abundance random effect values), alpha.star (the detection random effect values), and kappa. See Roberts and Rosenthal (2009) for details.

cov.model — a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".

NNGP — if TRUE, model is fit with an NNGP. See Datta et al. (2016) and Finley et al.

(2019) for more information. Currently only NNGP is supported, functionality for a Gaussian Process be addded in future package development.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.factors` the number of factors to use in the latent factor model approach. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and N (the number of species in the community).

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`transect` the type of transect. Currently supports line transects (`'line'`) or circular transects (i.e., point counts; `'point'`).

`det.func` the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal (`'halfnormal'`) and negative exponential (`'negexp'`).

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hypterthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems. Currently only relevant for spatial models.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report MCMC progress.

`n.burn` the number of samples out of the total `n.samples` to discard as burn-in for each chain. By default, the first 10% of samples is discarded.

n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class sfMsDS that is a list comprised of:

beta.comm.samples	a coda object of posterior samples for the community level abundance regression coefficients.
alpha.comm.samples	a coda object of posterior samples for the community level detection regression coefficients.
tau.sq.beta.samples	a coda object of posterior samples for the abundance community variance parameters.
tau.sq.alpha.samples	a coda object of posterior samples for the detection community variance parameters.
beta.samples	a coda object of posterior samples for the species level abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the species level detection regression coefficients.
kappa.samples	a coda object of posterior samples for the species level abundance dispersion parameters. Only included when family = 'NB'.
theta.samples	a coda object of posterior samples for the spatial correlation parameters for each spatial factor.
lambda.samples	a coda object of posterior samples for the spatial factor loadings.
w.samples	a three-dimensional array of posterior samples for the latent effects for each spatial factor. Array dimensions correspond to MCMC sample, spatial factor, then site.
N.samples	a three-dimensional array of posterior samples for the latent abundance values for each species. Note that these values always represent transect-level abundance, even when an offset is supplied. Array dimensions correspond to MCMC sample, species, then site.
mu.samples	a three-dimensional array of posterior samples for the latent expected abundance values for each species. When an offset is supplied in the data object, these correspond to expected abundance per unit area (i.e., density). Array dimensions correspond to MCMC sample, species, then site.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.

`sigma.sq.p.samples`
: a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples`
: a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`y.rep.samples`
: a four-dimensional array of fitted values. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`pi.samples`
: a four-dimensional array of cell-specific detection probabilities. Array dimensions correspond to MCMC samples, species, sites, and distance band.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

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

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Sollmann, R., Gardner, B., Williams, K. A., Gilbert, A. T., & Veit, R. R. (2016). A hierarchical distance sampling model to estimate abundance and covariate associations of species and communities. Methods in Ecology and Evolution, 7(5), 529-537.

Examples

```
set.seed(210)
J.x <- 10
J.y <- 10
```

```
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
family <- 'Poisson'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
factor.model <- TRUE
n.factors <- 3
phi <- runif(n.factors, 3 / 1, 3 / .2)
cov.model <- 'exponential'

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func,
               mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
               sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
               offset = offset, transect = transect, factor.model = factor.model,
               n.factors = n.factors)

y <- dat$y
X <- dat$X
X.p <- dat$X.p
coords <- dat$coords
dist.breaks <- dat$dist.breaks

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
```

```
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 10),
                   alpha.comm.normal = list(mean = 0, var = 10),
                   kappa.unif = list(0, 100),
                   phi.unif = list(3 / 1, 3 / .1),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
# Starting values
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, kappa = 1, phi = 3 / .5)

tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.8, lambda = 1, w = 1, phi = 0.8)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- sfMsDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
              det.formula = ~ det.cov.1,
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              family = 'Poisson',
              det.func = 'halfnormal',
              transect = transect,
              tuning = tuning,
              cov.model = 'exponential',
              NNGP = TRUE,
              n.neighbors = 5,
              n.factors = n.factors,
              priors = prior.list,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 10,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
summary(out, level = 'community')
```

sfMsNMix *Function for Fitting Spatial Factor Multi-species N-mixture Models*

Description

Function for fitting spatial multi-species N-mixture models with species correlations (i.e., an abundance-based spatially-explicit joint species distribution model with imperfect detection). We use Nearest Neighbor Gaussian Processes and a spatial factor modeling approach to achieve dimension reduction.

Usage

```
sfMsNMix(abund.formula, det.formula, data, inits, priors,
         tuning, cov.model = 'exponential', NNGP = TRUE, n.neighbors = 15,
        search.type = 'cb', n.factors, n.batch, batch.length, accept.rate = 0.43,
         family = 'Poisson', n.omp.threads = 1, verbose = TRUE, n.report = 100,
         n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
         n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, abund.covs, det.covs, coords, and offset. y is a three-dimensional array of observed count data with first dimension equal to the number of species, second dimension equal to the number of sites, and third dimension equal to the maximum number of replicates at a given site. abund.covs is a matrix or data frame containing the variables used in the abundance portion of the model, with J rows for each column (variable). det.covs is a list of variables included in the detection portion of the model. Each list element is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector of length J while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicates at a given site. coords is a matrix or data frame with two columns that contain the spatial coordinates of each site. Note that spAbundance assumes coordinates are specified in a projected coordinate system. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value or a vector with an offset for each site (e.g., if survey area differed in size).

inits — a list with each tag corresponding to a parameter name. Valid tags are alpha.comm, beta.comm, beta, alpha, tau.sq.beta, tau.sq.alpha, sigma.sq.mu, sigma.sq.p, phi, nu, lambda, w, kappa, and N. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the model, respectively. kappa is only relevant when family = 'NB'. nu is only relevant if cov.model = "matern". The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.

priors — a list with each tag corresponding to a parameter name. Valid tags are beta.comm.normal, alpha.comm.normal, tau.sq.beta.ig, tau.sq.alpha.ig, sigma.sq.mu.ig, sigma.sq.p.ig, kappa.unif, phi.unif, and nu.unif. Community-level abundance (beta.comm) and detection (alpha.comm) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances for the abundance coefficients are set to 100 and for the detection coefficients are set to 2.72. Community-level variance parameters for abundance (tau.sq.beta) and detection (tau.sq.alpha) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. sigma.sq.mu and sigma.sq.p are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. kappa is the negative binomial dispersion parameter for each species and is assumed to follow a uniform distribution. The hyperparameters of the uniform distribution are passed as a list of length two with first and second elements corresponding to the lower and upper bounds of the uniform distribution, respectively, which are each specified as vectors of length equal to the number of species or of length one if priors are the same for all species-specific dispersion parameters. The spatial factor model fits n.factors independent spatial processes. The spatial decay phi and smoothness nu parameters for each latent factor are assumed to follow Uniform distributions. The hyperparameters of the Uniform are passed as a list with two elements, with both elements being vectors of length n.factors corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factors. The priors for the factor loadings matrix lambda are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen

and Amemlya 2002). The upper triangular elements of the n.sp x n.factors matrix are fixed at 0 and the diagonal elements are fixed at 1. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1).

cov.model — a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".

tuning — a list with each tag corresponding to a parameter name, whose value defines the initial variance of the adaptive sampler. Valid tags are beta, alpha, beta.star (the abundance random effect values), alpha.star (the detection random effect values), phi, nu, lambda (the latent factor loadings), w (the latent factors), and kappa. See Roberts and Rosenthal (2009) for details.

NNGP — if TRUE, model is fit with an NNGP. See Datta et al. (2016) and Finley et al. (2019) for more information. Currently only NNGP is supported, functionality for a Gaussian Process may be added in future package development.

n.neighbors — number of neighbors used in the NNGP. Only used if NNGP = TRUE. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

search.type — a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: "cb" and "brute". The "cb" should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then "cb" and "brute" should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then "cb" and "brute" might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

n.factors — the number of factors to use in the spatial factor model approach. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and N (the number of species in the community).

n.batch — the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

batch.length — the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

accept.rate — target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

family — the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.

n.omp.threads — a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hypterthreaded cores. Note, n.omp.threads > 1 might not work on some systems. Currently only relevant for spatial models.

verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report MCMC progress.
n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class sfMsNMix that is a list comprised of:

beta.comm.samples
: a coda object of posterior samples for the community level abundance regression coefficients.

alpha.comm.samples
: a coda object of posterior samples for the community level detection regression coefficients.

tau.sq.beta.samples
: a coda object of posterior samples for the abundance community variance parameters.

tau.sq.alpha.samples
: a coda object of posterior samples for the detection community variance parameters.

beta.samples
: a coda object of posterior samples for the species level abundance regression coefficients.

alpha.samples
: a coda object of posterior samples for the species level detection regression coefficients.

lambda.samples
: a coda object of posterior samples for the spatial factor loadings.

theta.samples
: a coda object of posterior samples for the spatial correlation parameters for each spatial factor.

w.samples
: a three-dimensional array of posterior samples for the latent effects for each latent factor.

kappa.samples
: a coda object of posterior samples for the species level abundance dispersion parameters. Only included when family = 'NB'.

N.samples
: a three-dimensional array of posterior samples for the latent abundance values for each species.

mu.samples
: a three-dimensional array of posterior samples for the latent expected abundance values for each species.

sigma.sq.mu.samples
: a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.

`sigma.sq.p.samples`
: a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples`
: a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation. Note that detection probability estimated values are not included in the model object, but can be extracted using `fitted()`.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Christensen, W. F., and Amemiya, Y. (2002). Latent variable analysis of multivariate spatial data. *Journal of the American Statistical Association*, 97(457), 302-317.

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

Yamaura, Y., Royle, J. A., Shimada, N., Asanuma, S., Sato, T., Taki, H., & Makino, S. I. (2012). Biodiversity of man-made open habitats in an underused country: a class of multispecies abundance models for count data. Biodiversity and Conservation, 21(6), 1365-1380.

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
```

```
n.rep <- sample(5, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
# Abundance
beta.mean <- c(0, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Detection
alpha.mean <- c(0, 0.5, 0.8)
tau.sq.alpha <- c(0.2, 1, 1.5)
p.det <- length(alpha.mean)
# Random effects
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 3
phi <- runif(n.factors, 3 / 1, 3 / .2)

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta, alpha = alpha,
                 mu.RE = mu.RE, p.RE = p.RE, family = 'Poisson',
                 factor.model = TRUE, n.factors = n.factors, sp = TRUE, phi = phi,
                 cov.model = 'exponential')

y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.re <- dat$X.re
X.p.re <- dat$X.p.re
coords <- dat$coords

# Package all data into a list
abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')
det.covs <- list(det.cov.1 = as.data.frame(X.p[, , 2]),
                 det.cov.2 = as.data.frame(X.p[, , 3]))
data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)
prior.list <- list(beta.comm.normal = list(mean = rep(0, p.abund),
                                           var = rep(100, p.abund)),
                   alpha.comm.normal = list(mean = rep(0, p.det),
                                            var = rep(2.72, p.det)),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = 0.1, b = 0.1),
```

```
                  phi.unif = list(a = 3 / 1, 3 / .1))
inits.list <- list(beta.comm = 0, alpha.comm = 0,
                  beta = 0, alpha = 0,
                  tau.sq.beta = 0.5, tau.sq.alpha = 0.5,
                  N = apply(y, c(1, 2), max, na.rm = TRUE))
tuning.list <- list(beta = 0.5, alpha = 0.5, lambda = 0.5, w = 0.5,
                   phi = 1)

n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- sfMsNMix(abund.formula = ~ abund.cov.1,
                det.formula = ~ det.cov.1 + det.cov.2,
                data = data.list,
                n.batch = n.batch,
                inits = inits.list,
                priors = prior.list,
                tuning = tuning.list,
                batch.length = batch.length,
                n.omp.threads = 1,
                n.factors = n.factors,
                cov.model = 'exponential',
                n.neighbors = 5,
                verbose = TRUE,
                n.report = 1,
                n.burn = n.burn,
                n.thin = n.thin,
                n.chains = n.chains)

summary(out, level = 'community')
```

---

simAbund *Simulate Univariate Data for Testing GLMMs*

Description

The function simAbund simulates univariate data without imperfect detection for simulation studies, power assessments, or function testing related to GLMMs. Data can be optionally simulated with a spatial Gaussian Process in the model. Non-spatial random effects can also be included in the model.

Usage

```
simAbund(J.x, J.y, n.rep, n.rep.max, beta, kappa, tau.sq, mu.RE = list(),
         offset = 1, sp = FALSE, svc.cols = 1, cov.model, sigma.sq, phi, nu,
         family = 'Poisson', z, x.positive = FALSE, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.
J.y	a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
n.rep	a numeric vector of length $J = J.x \times J.y$ indicating the number of replicate surveys at each of the J sites.
n.rep.max	a single numeric value indicating the maximum number of replicate surveys. This is an optional argument, with its default value set to max(n.rep). This can be used to generate data sets with different types of missingness (e.g., simulate data across 20 days (replicate surveys) but sites are only sampled a maximum of ten times each).
beta	a numeric vector containing the intercept and regression coefficient parameters for the abundance model.
kappa	a single numeric value containing the dispersion parameter for the abundance portion of the model. Only relevant when family = 'NB'.
tau.sq	a single numeric value containing the residual variance parameter of the Gaussian distribution. Only relevant when family = 'Gaussian' or family = 'zi-Gaussian'.
mu.RE	a list used to specify the non-spatial random intercepts included in the model. The list must have two tags: levels and sigma.sq.mu. levels is a vector of length equal to the number of distinct random intercepts to include in the model and contains the number of levels there are in each intercept. sigma.sq.mu is a vector of length equal to the number of distinct random intercepts to include in the model and contains the variances for each random effect. A third optional tage is beta.indx, which is a numeric vector with length equal to the number of distinct random intercepts. The values in beta.indx denote the intercept/covariate for which you wish to simulate a random intercept/slope. Numeric values correspond to the intercept/covaraite in beta. If mu.RE is not specified, no random effects are included in the abundance portion of the model.
sp	a logical value indicating whether to simulate a spatially-explicit model with a Gaussian process. By default set to FALSE.
offset	either a single numeric value, a vector of length J, or a site by replicate matrix that contains the offset for each data point in the data set.
svc.cols	a vector indicating the variables whose effects will be estimated as spatially-varying coefficients. svc.cols is an integer vector with values indicating the order of covariates specified in the model formula (with 1 being the intercept if specified).
cov.model	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the abundance data. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
sigma.sq	a numeric value indicating the spatial variance parameter. Ignored when sp = FALSE.
phi	a numeric value indicating the spatial decay parameter. Ignored when sp = FALSE.

nu	a numeric value indicating the spatial smoothness parameter. Only used when sp = TRUE and cov.model = "matern".
family	the distribution to use for the data. Currently supports 'NB' (negative binomial), 'Poisson', 'Gaussian', and 'zi-Gaussian'.
z	a vector of length J containing the binary presence/absence portion of a zero-inflated Gaussian model. Only relevant when family = 'zi-Gaussian'.
x.positive	a logical value indicating whether the simulated covariates should be simulated as random standard normal covariates (x.positive = FALSE) or restricted to positive values using a uniform distribution with lower bound 0 and upper bound 1 (x.positive = TRUE).
...	currently no additional arguments

Value

A list comprised of:

X	a three-dimensional numeric design array of covariates with dimensions corresponding to sites, replicates, and number of covariates (including an intercept) for the model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.
w	a matrix of the spatial random effects. Only used to simulate data when sp = TRUE. If simulating data with spatially-varying coefficients, the number of columns equals the number of spatially-varying coefficients and each row corresponds to a site.
mu	a J x max(n.rep) matrix of the expected abundance values for each site and replicate survey.
y	a J x max(n.rep) matrix of the raw count data for each site and replicate combination.
X.re	a numeric three-dimensional array containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
beta.star	a numeric vector that contains the simulated abundance random effects for each given level of the random effects included in the abundance model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
set.seed(401)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
```

```
beta <- c(0, -1.5, 0.3, -0.8)
p.abund <- length(beta)
mu.RE <- list(levels = c(30), sigma.sq.mu = c(1.3))
kappa <- 0.5
sp <- FALSE
family <- 'NB'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, family = 'NB')
```

`simDS` *Simulate Single-Species Distance Sampling Data*

Description

The function `simDS` simulates single-species distance sampling data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the abundance portion of the model. Non-spatial random effects can also be included in the detection or abundance portions of the distance sampling model.

Usage

```
simDS(J.x, J.y, n.bins, bin.width, beta, alpha, det.func, transect = 'line',
      kappa, mu.RE = list(), p.RE = list(), offset = 1,
      sp = FALSE, cov.model, sigma.sq, phi, nu, family = 'Poisson', ...)
```

Arguments

`J.x`	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.
`J.y`	a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
`n.bins`	a single numeric value indicating the number of distance bins from which to generate data.
`bin.width`	a vector of length `n.bins` indicating the length of each bin. Lengths can be different for each distance bin or the same across bins.
`beta`	a numeric vector containing the intercept and regression coefficient parameters for the abundance portion of the single-species distance sampling model.
`alpha`	a numeric vector containing the intercept and regression coefficient parameters for the detection portion of the single-species distance sampling model.
`det.func`	the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal (`'halfnormal'`) and negative exponential (`'negexp'`).
`transect`	the type of transect. Currently supports line transects (`'line'`) or circular transects (i.e., point counts; `'point'`).

kappa	a single numeric value containing the dispersion parameter for the abundance portion of the hierarchical distance sampling model. Only relevant when family = 'NB'.
mu.RE	a list used to specify the non-spatial random intercepts included in the abundance portion of the model. The list must have two tags: levels and sigma.sq.mu. levels is a vector of length equal to the number of distinct random intercepts to include in the model and contains the number of levels there are in each intercept. sigma.sq.mu is a vector of length equal to the number of distinct random intercepts to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the abundance portion of the model.
p.RE	a list used to specify the non-spatial random intercepts included in the detection portion of the model. The list must have two tags: levels and sigma.sq.p. levels is a vector of length equal to the number of distinct random intercepts to include in the model and contains the number of levels there are in each intercept. sigma.sq.p is a vector of length equal to the number of distinct random intercepts to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the detection portion of the model.
offset	either a single numeric value or a vector of length J that contains the offset for each location in the data set.
sp	a logical value indicating whether to simulate a spatially-explicit HDS model with a Gaussian process. By default set to FALSE.
cov.model	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the latent abundance values. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
sigma.sq	a numeric value indicating the spatial variance parameter. Ignored when sp = FALSE.
phi	a numeric value indicating the spatial decay parameter. Ignored when sp = FALSE.
nu	a numeric value indicating the spatial smoothness parameter. Only used when sp = TRUE and cov.model = "matern".
family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times p.abund$ numeric design matrix for the abundance portion of the model.
X.p	a $J \times p.abund$ numeric design matrix for the detection portion of the model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.

w	a $J \times 1$ matrix of the spatial random effects. Only used to simulate data when sp = TRUE.
mu	a $J \times 1$ matrix of the expected abundance values for each site.
N	a length J vector of the latent abundances at each site.
p	a length J vector of the detection probabilities at each site.
pi.full	a J x n.bins + 1 vector of the cell-specific detection probabilities for each site, where the last column indicates the probability of not detecting an individual at that site.
y	a J x max(n.bins) matrix of the raw count data for each site and distance bin.
X.p.re	a numeric matrix containing the levels of any detection random effect included in the model. Only relevant when detection random effects are specified in p.RE.
X.re	a numeric matrix containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
alpha.star	a numeric vector that contains the simulated detection random effects for each given level of the random effects included in the detection model. Only relevant when detection random effects are included in the model.
beta.star	a numeric vector that contains the simulated abundance random effects for each given level of the random effects included in the HDS model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

Examples

```
set.seed(110)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Abundance coefficients
beta <- c(1.0, 0.2, 0.3, -0.2)
p.abund <- length(beta)
# Detection coefficients
alpha <- c(-1.0, -0.3)
p.det <- length(alpha)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
sp <- FALSE
family <- 'NB'
```

```
kappa <- 0.1
offset <- 1.8
transect <- 'point'

dat <- simDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
             beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
             mu.RE = mu.RE, p.RE = p.RE, sp = sp,
             sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
             offset = offset, transect = transect)
```

`simMsAbund` *Simulate Multivariate Data for Testing GLMMs*

Description

The function `simMsAbund` simulates multivariate data without imperfect detection for simulation studies, power assessments, or function testing related to GLMMs. Data can be optionally simulated with a spatial Gaussian Process in the model, as well as an option to allow for species correlations using a factor modeling approach. Non-spatial random effects can also be included in the abundance portions of the model.

Usage

```
simMsAbund(J.x, J.y, n.rep, n.rep.max, n.sp, beta, kappa, tau.sq, mu.RE = list(),
           offset = 1, sp = FALSE, cov.model, svc.cols = 1,
           sigma.sq, phi, nu, family = 'Poisson',
           factor.model = FALSE, n.factors, z, ...)
```

Arguments

`J.x`	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.
`J.y`	a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
`n.rep`	a numeric vector of length $J = J.x \times J.y$ indicating the number of replicate surveys at each of the J sites.
`n.rep.max`	a single numeric value indicating the maximum number of replicate surveys. This is an optional argument, with its default value set to `max(n.rep)`. This can be used to generate data sets with different types of missingness (e.g., simulate data across 20 days (replicate surveys) but sites are only sampled a maximum of ten times each).
`n.sp`	a single numeric value indicating the number of species to simulate count data.
`beta`	a numeric matrix with `n.sp` rows containing the intercept and regression coefficient parameters for the model. Each row corresponds to the regression coefficients for a given species.

kappa	a numeric vector of length n.sp containing the dispersion parameter for the model for each species. Only relevant when family = 'NB'.
tau.sq	a numeric vector of length n.sp containing the residual variance parameters for the model for each species. Only relevant for Gaussian or zero-inflated Gaussian models.
mu.RE	a list used to specify the non-spatial random intercepts included in the model. The list must have two tags: levels and sigma.sq.mu. levels is a vector of length equal to the number of distinct random intercepts to include in the model and contains the number of levels there are in each intercept. sigma.sq.mu is a vector of length equal to the number of distinct random intercepts to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the model.
offset	either a single numeric value, a vector of length J, or a site by replicate matrix that contains the offset for each data point in the data set.
sp	a logical value indicating whether to simulate a spatially-explicit model with a Gaussian process. By default set to FALSE.
cov.model	a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the abundance values. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
svc.cols	a vector indicating the variables whose effects will be estimated as spatially-varying coefficients. svc.cols is an integer vector with values indicating the order of covariates specified in the model formula (with 1 being the intercept if specified).
sigma.sq	a numeric vector of length n.sp containing the spatial variance parameter for each species. Ignored when sp = FALSE or when factor.model = TRUE.
phi	a numeric vector of length n.sp containing the spatial decay parameter for each species. Ignored when sp = FALSE. If factor.model = TRUE, this should be of length n.factors.
nu	a numeric vector of length n.sp containing the spatial smoothness parameter for each species. Only used when sp = TRUE and cov.model = 'matern'. If factor.model = TRUE, this should be of length n.factors.
factor.model	a logical value indicating whether to simulate data following a factor modeling approach that explicitly incoporates species correlations. If sp = TRUE, the latent factors are simulated from independent spatial processes. If sp = FALSE, the latent factors are simulated from standard normal distributions.
n.factors	a single numeric value specifying the number of latent factors to use to simulate the data if factor.model = TRUE.
family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
z	a matrix with n.sp rows and J columns containing the binary presence/absence portion of a zero-inflated Gaussian model for each species. Only relevant when family = 'zi-Gaussian'.
...	currently no additional arguments

Value

A list comprised of:

X	a three-dimensional numeric design array of covariates with dimensions corresponding to sites, replicates, and number of covariates (including an intercept) for the model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.
w	a list of $N \times J$ matrices of the spatially-varying coefficients for each species. Each element of the list corresponds to a different spatially-varying coefficient. Only used to simulate data when sp = TRUE. If factor.model = TRUE, the first dimension of each matrix is n.factors.
mu	a n.sp x J matrix of the mean abundances for each species at each site.
y	a n.sp x J x max(n.rep) array of the raw count data for each species at each site and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
X.re	a numeric matrix containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
beta.star	a numeric matrix where each row contains the simulated abundance random effects for each given level of the random effects included in the abundance model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
set.seed(408)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, size = J, replace = TRUE)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(-2, 0.5)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2)
# Random effects (two random intercepts)
mu.RE <- list(levels = c(10, 15),
              sigma.sq.mu = c(0.43, 0.5))
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- TRUE
```

```
n.factors <- 2
factor.model <- TRUE
phi <- runif(n.factors, 3/1, 3 / .1)
kappa <- runif(n.sp, 0.1, 1)

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, kappa = kappa, family = 'NB',
                  factor.model = factor.model, phi = phi,
                  cov.model = 'spherical', n.factors = n.factors)
```

`simMsDS` *Simulate Multi-Species Distance Sampling Data*

Description

The function `simMsDS` simulates multi-species distance sampling data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the abundance portion of the model, as well as an option to allow for species correlations using a factor modeling approach. Non-spatial random effects can also be included in the detection or abundance portions of the model.

Usage

```
simMsDS(J.x, J.y, n.bins, bin.width, n.sp, beta, alpha,
        det.func, transect = 'line', kappa, mu.RE = list(),
        p.RE = list(), offset = 1, sp = FALSE, cov.model,
        sigma.sq, phi, nu, family = 'Poisson',
        factor.model = FALSE, n.factors, ...)
```

Arguments

`J.x`	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.
`J.y`	a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
`n.bins`	a single numeric value indicating the number of distance bins from which to generate data.
`bin.width`	a vector of length `n.bins` indicating the length of each bin. Lengths can be different for each distance bin or the same across bins.
`n.sp`	a single numeric value indicating the number of species to simulate count data.
`beta`	a numeric matrix with `n.sp` rows containing the intercept and regression coefficient parameters for the abundance portion of the multi-species hierarchical distance sampling (HDS) model. Each row corresponds to the regression coefficients for a given species.
`alpha`	a numeric matrix with `n.sp` rows containing the intercept and regression coefficient parameters for the detection portion of the multi-species HDS model. Each row corresponds to the regression coefficients for a given species.

`det.func` the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal (`'halfnormal'`) and negative exponential (`'negexp'`).

`transect` the type of transect. Currently supports line transects (`'line'`) or circular transects (i.e., point counts; `'point'`).

`kappa` a numeric vector of length `n.sp` containing the dispersion parameter for the abundance portion of the HDS model for each species. Only relevant when `family = 'NB'`.

`mu.RE` a list used to specify the non-spatial random effects included in the abundance portion of the model. The list must have two tags: `levels` and `sigma.sq.mu`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effect. `sigma.sq.mu` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the abundance portion of the model. An optional third tag, `beta.indx`, is a list that contains integers denoting the corresponding value of `beta` that each random effect corresponds to. This allows specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

`p.RE` a list used to specify the non-spatial random effects included in the detection portion of the model. The list must have two tags: `levels` and `sigma.sq.p`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effects. `sigma.sq.p` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the detection portion of the model. An optional third tag, `alpha.indx`, is a list that contains integers denoting the corresponding value of `alpha` that each random effect corresponds to. This allows specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

`offset` either a single numeric value or a vector of length `J` that contains the offset for each location in the data set.

`sp` a logical value indicating whether to simulate a spatially-explicit model with a Gaussian process. By default set to `FALSE`.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the latent abundance values. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`sigma.sq` a numeric vector of length `n.sp` containing the spatial variance parameter for each species. Ignored when `sp = FALSE` or when `factor.model = TRUE`.

`phi` a numeric vector of length `n.sp` containing the spatial decay parameter for each species. Ignored when `sp = FALSE`. If `factor.model = TRUE`, this should be of length `n.factors`.

`nu` a numeric vector of length `n.sp` containing the spatial smoothness parameter for each species. Only used when `sp = TRUE` and `cov.model = 'matern'`. If `factor.model = TRUE`, this should be of length `n.factors`.

factor.model	a logical value indicating whether to simulate data following a factor modeling approach that explicitly incoporates species correlations. If sp = TRUE, the latent factors are simulated from independent spatial processes. If sp = FALSE, the latent factors are simulated from standard normal distributions.
n.factors	a single numeric value specifying the number of latent factors to use to simulate the data if factor.model = TRUE.
family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times p.abund$ numeric design matrix for the abundance portion of the model.
X.p	a $J \times p.abund$ numeric design matrix for the detection portion of the model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.
w	a $N \times J$ matrix of the spatial random effects for each species. Only used to simulate data when sp = TRUE. If factor.model = TRUE, the first dimension is n.factors.
mu	a n.sp x J matrix of the expected abundances for each species at each site.
N	a n.sp x J matrix of the latent occurrence states for each species at each site.
p	a n.sp x J x max(n.rep) array of the detection probabilities for each species at each site and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
y	a n.sp x J x max(n.rep) array of the raw distance sampling data for each species at each site and and distance bin.
X.p.re	a numeric matrix containing the levels of any detection random effect included in the model. Only relevant when detection random effects are specified in p.RE.
X.re	a numeric matrix containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
alpha.star	a numeric matrix where each row contains the simulated detection random effects for each given level of the random effects included in the detection model. Only relevant when detection random effects are included in the model.
beta.star	a numeric matrix where each row contains the simulated abundance random effects for each given level of the random effects included in the abundance model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Number of species
n.sp <- 5
# Community-level abundance coefficients
beta.mean <- c(-1, 0.2, 0.3, -0.2)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 0.3, 0.5, 0.4)
# Detection coefficients
alpha.mean <- c(-1.0, -0.3)
p.det <- length(alpha.mean)
tau.sq.alpha <- c(0.1, 0.2)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE
family <- 'NB'
kappa <- runif(n.sp, 0.3, 3)
offset <- pi * .8^2
transect <- 'line'
factor.model <- FALSE

dat <- simMsDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
               n.sp = n.sp, beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
                 mu.RE = mu.RE, p.RE = p.RE, sp = sp, cov.model = cov.model,
                 sigma.sq = sigma.sq, phi = phi, nu = nu, family = family,
                 offset = offset, transect = transect, factor.model = factor.model)
```

`simMsNMix` *Simulate Multi-Species Repeated Count Data with Imperfect Detection*

Description

The function `simMsNMix` simulates multi-species count data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the abundance portion of the model, as well as an option to allow for species correlations using a factor modeling approach. Non-spatial random effects can also be included in the detection or abundance portions of the model.

Usage

```
simMsNMix(J.x, J.y, n.rep, n.rep.max, n.sp, beta, alpha, kappa, mu.RE = list(),
          p.RE = list(), offset = 1, sp = FALSE, cov.model,
          sigma.sq, phi, nu, family = 'Poisson',
          factor.model = FALSE, n.factors, ...)
```

Arguments

`J.x`	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.
`J.y`	a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
`n.rep`	a numeric vector of length $J = J.x \times J.y$ indicating the number of repeat visits at each of the J sites.
`n.rep.max`	a single numeric value indicating the maximum number of replicate surveys. This is an optional argument, with its default value set to `max(n.rep)`. This can be used to generate data sets with different types of missingness (e.g., simulate data across 20 days (replicate surveys) but sites are only sampled a maximum of ten times each).
`n.sp`	a single numeric value indicating the number of species to simulate count data.
`beta`	a numeric matrix with `n.sp` rows containing the intercept and regression coefficient parameters for the abundance portion of the multi-species N-mixture model. Each row corresponds to the regression coefficients for a given species.
`alpha`	a numeric matrix with `n.sp` rows containing the intercept and regression coefficient parameters for the detection portion of the multi-species N-mixture model. Each row corresponds to the regression coefficients for a given species.
`kappa`	a numeric vector of length `n.sp` containing the dispersion parameter for the abundance portion of the N-mixture model for each species. Only relevant when `family = 'NB'`.
`mu.RE`	a list used to specify the non-spatial random effects included in the abundance portion of the model. The list must have two tags: `levels` and `sigma.sq.mu`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effect. `sigma.sq.mu` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the abundance portion of the model. An optional third tag, `beta.indx`, is a list that contains integers denoting the corresponding value of `beta` that each random effect corresponds to. This allows

specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

`p.RE` a list used to specify the non-spatial random effects included in the detection portion of the model. The list must have two tags: `levels` and `sigma.sq.p`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effects. `sigma.sq.p` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the detection portion of the model. An optional third tag, `alpha.indx`, is a list that contains integers denoting the corresponding value of `alpha` that each random effect corresponds to. This allows specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

`offset` either a single numeric value or a vector of length `J` that contains the offset for each location in the data set.

`sp` a logical value indicating whether to simulate a spatially-explicit model with a Gaussian process. By default set to `FALSE`.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the latent abundance values. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`sigma.sq` a numeric vector of length `n.sp` containing the spatial variance parameter for each species. Ignored when `sp = FALSE` or when `factor.model = TRUE`.

`phi` a numeric vector of length `n.sp` containing the spatial decay parameter for each species. Ignored when `sp = FALSE`. If `factor.model = TRUE`, this should be of length `n.factors`.

`nu` a numeric vector of length `n.sp` containing the spatial smoothness parameter for each species. Only used when `sp = TRUE` and `cov.model = 'matern'`. If `factor.model = TRUE`, this should be of length `n.factors`.

`factor.model` a logical value indicating whether to simulate data following a factor modeling approach that explicitly incoporates species correlations. If `sp = TRUE`, the latent factors are simulated from independent spatial processes. If `sp = FALSE`, the latent factors are simulated from standard normal distributions.

`n.factors` a single numeric value specifying the number of latent factors to use to simulate the data if `factor.model = TRUE`.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`...` currently no additional arguments

Value

A list comprised of:

`X` a $J \times p.abund$ numeric design matrix for the abundance portion of the model.

X.p	a three-dimensional numeric array with dimensions corresponding to sites, repeat visits, and number of detection regression coefficients. This is the design matrix used for the detection portion of the N-mixture model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.
w	a $N \times J$ matrix of the spatial random effects for each species. Only used to simulate data when sp = TRUE. If factor.model = TRUE, the first dimension is n.factors.
mu	a n.sp x J matrix of the expected abundances for each species at each site.
N	a n.sp x J matrix of the latent occurrence states for each species at each site.
p	a n.sp x J x max(n.rep) array of the detection probabilities for each species at each site and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
y	a n.sp x J x max(n.rep) array of the raw count data for each species at each site and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
X.p.re	a three-dimensional numeric array containing the levels of any detection random effect included in the model. Only relevant when detection random effects are specified in p.RE.
X.re	a numeric matrix containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
alpha.star	a numeric matrix where each row contains the simulated detection random effects for each given level of the random effects included in the detection model. Only relevant when detection random effects are included in the model.
beta.star	a numeric matrix where each row contains the simulated abundance random effects for each given level of the random effects included in the abundance model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

Examples

```
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
n.sp <- 10
# Community-level covariate effects
# Abundance
beta.mean <- c(0.2, -0.15)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2)
```

```
tau.sq.alpha <- c(0.2, 0.3)
p.det <- length(alpha.mean)
mu.RE <- list(levels = c(10, 12),
              sigma.sq.mu = c(1.5, 0.3),
              beta.indx = list(1, 2))
p.RE <- list(levels = c(15, 10),
             sigma.sq.p = c(0.8, 0.5),
             alpha.indx = list(1, 2))
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
alpha <- matrix(NA, nrow = n.sp, ncol = p.det)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(n.sp, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
factor.model <- TRUE
n.factors <- 3
# Spatial parameters if desired
phi <- runif(n.factors, 3/1, 3/.1)
sp <- TRUE
family <- 'Poisson'

dat <- simMsNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                 alpha = alpha, mu.RE = mu.RE, p.RE = p.RE, sp = TRUE,
                 cov.model = 'exponential', phi = phi, factor.model = factor.model,
                 n.factors = n.factors, family = family)
```

`simNMix` *Simulate Single-Species Count Data with Imperfect Detection*

Description

The function `simNMix` simulates single-species count data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the abundance portion of the model. Non-spatial random intercepts/slopes can also be included in the detection or abundance portions of the N-mixture model.

Usage

```
simNMix(J.x, J.y, n.rep, n.rep.max, beta, alpha, kappa, mu.RE = list(),
        p.RE = list(), offset = 1, sp = FALSE, cov.model, sigma.sq, phi, nu,
        family = 'Poisson', ...)
```

Arguments

`J.x`	a single numeric value indicating the number of sites to simulate count data along the horizontal axis. Total number of sites with simulated data is $J.x \times J.y$.

J.y — a single numeric value indicating the number of sites to simulate count data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.

n.rep — a numeric vector of length $J = J.x \times J.y$ indicating the number of repeat visits at each of the J sites.

n.rep.max — a single numeric value indicating the maximum number of replicate surveys. This is an optional argument, with its default value set to `max(n.rep)`. This can be used to generate data sets with different types of missingness (e.g., simulate data across 20 days (replicate surveys) but sites are only sampled a maximum of ten times each).

beta — a numeric vector containing the intercept and regression coefficient parameters for the abundance portion of the single-species N-mixture model.

alpha — a numeric vector containing the intercept and regression coefficient parameters for the detection portion of the single-species N-mixture model.

kappa — a single numeric value containing the dispersion parameter for the abundance portion of the N-mixture model. Only relevant when `family = 'NB'`.

mu.RE — a list used to specify the non-spatial random effects included in the abundance portion of the model. The list must have two tags: `levels` and `sigma.sq.mu`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effect. `sigma.sq.mu` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the abundance portion of the model. An optional third tag, `beta.indx`, is a list that contains integers denoting the corresponding value of `beta` that each random effect corresponds to. This allows specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

p.RE — a list used to specify the non-spatial random effects included in the detection portion of the model. The list must have two tags: `levels` and `sigma.sq.p`. `levels` is a vector of length equal to the number of distinct random effects to include in the model and contains the number of levels there are in each effects. `sigma.sq.p` is a vector of length equal to the number of distinct random effects to include in the model and contains the variances for each random effect. If not specified, no random effects are included in the detection portion of the model. An optional third tag, `alpha.indx`, is a list that contains integers denoting the corresponding value of `alpha` that each random effect corresponds to. This allows specification of random intercepts as well as slopes. By default, all effects are assumed to be random intercepts.

offset — either a single numeric value or a vector of length `J` that contains the offset for each location in the data set.

sp — a logical value indicating whether to simulate a spatially-explicit N-mixture model with a Gaussian process. By default set to `FALSE`.

cov.model — a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the latent abundance values. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

sigma.sq	a numeric value indicating the spatial variance parameter. Ignored when sp = FALSE.
phi	a numeric value indicating the spatial decay parameter. Ignored when sp = FALSE.
nu	a numeric value indicating the spatial smoothness parameter. Only used when sp = TRUE and cov.model = "matern".
family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times p.abund$ numeric design matrix for the abundance portion of the model.
X.p	a three-dimensional numeric array with dimensions corresponding to sites, repeat visits, and number of detection regression coefficients. This is the design matrix used for the detection portion of the N-mixture model.
coords	a $J \times 2$ numeric matrix of coordinates of each site. Required for spatial models.
w	a $J \times 1$ matrix of the spatial random effects. Only used to simulate data when sp = TRUE.
mu	a $J \times 1$ matrix of the expected abundance values for each site.
N	a length J vector of the latent abundances at each site.
p	a J x max(n.rep) matrix of the detection probabilities for each site and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
y	a J x max(n.rep) matrix of the raw count data for each site and replicate combination.
X.p.re	a three-dimensional numeric array containing the levels of any detection random effect included in the model. Only relevant when detection random effects are specified in p.RE.
X.re	a numeric matrix containing the levels of any abundance random effect included in the model. Only relevant when abundance random effects are specified in mu.RE.
alpha.star	a numeric vector that contains the simulated detection random effects for each given level of the random effects included in the detection model. Only relevant when detection random effects are included in the model.
beta.star	a numeric vector that contains the simulated abundance random effects for each given level of the random effects included in the N-mixture model. Only relevant when abundance random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
set.seed(400)
J.x <- 10
J.y <- 10
n.rep <- rep(4, J.x * J.y)
beta <- c(0.5, -0.15)
alpha <- c(0.7, 0.4)
kappa <- 0.5
phi <- 3 / .6
sigma.sq <- 2
mu.RE <- list(levels = 10, sigma.sq.mu = 1.2)
p.RE <- list(levels = 15, sigma.sq.p = 0.8)
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               kappa = kappa, mu.RE = mu.RE, p.RE = p.RE, sp = TRUE,
               cov.model = 'spherical', sigma.sq = sigma.sq, phi = phi,
               family = 'NB')
```

---

spAbund *Function for Fitting Univariate Spatial Abundance GLMs*

Description

The function spAbund fits univariate spatial abundance GLMs.

Usage

```
spAbund(formula, data, inits, priors, tuning,
        cov.model = 'exponential', NNGP = TRUE,
        n.neighbors = 15, search.type = 'cb',
        n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
        n.omp.threads = 1, verbose = TRUE, n.report = 100,
        n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
        n.chains = 1, save.fitted = TRUE, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, z, coords, and offset. y is a vector, matrix, or data frame of the observed count values. If a vector, the values represent the observed counts at each site. If multiple replicate observations are obtained at the sites (e.g., sub-samples, repeated sampling over multiple seasons), y can be specified as a matrix or data frame with first dimension equal to the number of sites (J) and second dimension equal to the maximum number of replicates at a given site. covs is either a data frame or list containing the variables used in the model. When only fitting a model with site-level data, covs can be specified as a data frame, with

each row corresponding to site and each column corresponding to a variable. When multiple abundance values are available at a site, covs is specified as a list, where each list element is a different covariate, which can be site-level or observation-level. Site-level covariates are specified as a vector of length J, while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicate observations at a given site. coords is a $J \times 2$ matrix of the observation coordinates. Note that spAbundance assumes coordinates are specified in a projected coordinate system. For zero-inflated Gaussian models, the tag z is used to specify the binary component of the zero-inflated model and should have the same length as y. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value, a vector with an offset for each site (e.g., if survey area differed in size), or a site x replicate matrix if more than one count is available at a given site.

inits — a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, phi, w, nu, kappa, sigma.sq.mu, tau.sq. nu is only specified if cov.model = "matern", sigma.sq.mu is only specified if there are random effects in formula, and kappa is only specified when family = 'NB'. tau.sq is only specified when family = 'Gaussian' or family = 'zi-Gaussian'. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.

priors — a list with each tag corresponding to a parameter name. Valid tags are beta.normal, phi.unif, sigma.sq.ig, nu.unif, kappa.unif, sigma.sq.mu.ig, tau.sq.ig. Abundance (beta) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances are set to 100. The spatial variance parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution. The spatial decay phi, spatial smoothness nu, and negative binomial dispersion kappa parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for sigma.sq are passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively. sigma.sq.mu are the random effect variances for any random effects, and are assumed to follow an inverse-Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances. tau.sq is the residual variance for Gaussian (or zero-inflated Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a vector of length two, with the first and

second element corresponding to the shape and scale parameters, respectively.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `beta`, `alpha`, `beta.star` (the abundance random effect values), `kappa`, `phi`, and `nu`. See Roberts and Rosenthal (2009) for details. Note that only `phi` and `nu` are the only parameters that require tuning for a Gaussian or zero-inflated Gaussian model.

`NNGP` if `TRUE`, model is fit with an NNGP. See Datta et al. (2016) and Finley et al. (2019) for more information. Currently only NNGP is supported, functionality for a full GP may be addded in future package development.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial), `'Poisson'`, `'Gaussian'`, and `'zi-Gaussian'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of hyperthreaded cores. Note, `n.omp.threads` > 1 might not work on some systems.

`verbose` if `TRUE`, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.

`n.report` the interval to report Metropolis sampler acceptance and MCMC progress.

`n.burn` the number of samples out of the total `n.batch * batch.length` samples in each chain to discard as burn-in. By default, the first 10% of samples is discarded.

n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of MCMC chains to run in sequence.
save.fitted	logical value indicating whether or not fitted values and likelihood values should be saved in the resulting model object. If save.fitted = FALSE, the components y.rep.samples, mu.samples, and like.samples will not be included in the model object, and subsequent functions for calculating WAIC, fitted values, and posterior predictive checks will not work, although they all can be calculated manually if desired. Setting save.fitted = FALSE can be useful when working with very large data sets to minimize the amount of RAM needed when fitting and storing the model object in memory.
...	currently no additional arguments

Value

An object of class spAbund that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance dispersion parameter. Only included when family = 'NB'.
tau.sq.samples	a coda object of posterior samples for the Gaussian residual variance parameter. Only included when family = 'Gaussian' or family = 'zi-Gaussian'.
y.rep.samples	a two or three-dimensional object of posterior samples for the abundance replicate (fitted) values with dimensions corresponding to MCMC samples, site, and replicate.
mu.samples	a two or -three-dimensional array of posterior samples for the expected abundance samples with dimensions corresponding to MCMC samples, site, and replicate.
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a coda object of posterior samples for latent spatial random effects.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the model. Only included if random effects are specified in formula.
beta.star.samples	a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in formula.
like.samples	a coda object of posterior samples for the likelihood value associated with each site. Used for calculating WAIC.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Examples

```
set.seed(888)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0, -1.5, 0.3, -0.8)
p.abund <- length(beta)
mu.RE <- list(levels = c(50, 45),
              sigma.sq.mu = c(1.3, 0.5),
              beta.indx = c(1, 2))
phi <- 3/.6
sigma.sq <- 2
kappa <- 0.2
sp <- TRUE
cov.model <- 'exponential'
family <- 'NB'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, phi = phi,
                sigma.sq = sigma.sq, cov.model = cov.model, family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re
coords <- dat$coords

covs <- list(int = X[, , 1],
             abund.cov.1 = X[, , 2],
             abund.cov.2 = X[, , 3],
             abund.cov.3 = X[, , 4],
             abund.factor.1 = X.re[, , 1],
             abund.factor.2 = X.re[, , 2])
```

```
data.list <- list(y = y, covs = covs, coords = coords)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 100),
                   phi.unif = c(3 / 1, 3 / .1),
                   sigma.sq.ig = c(2, 1),
                   kappa.unif = c(0.001, 10))
# Starting values
inits.list <- list(beta = beta, kappa = kappa, sigma.sq = sigma.sq, phi = phi)

tuning <- list(phi = 0.3, kappa = 0.05, beta = 0.1, beta.star = 0.1, w = 0.1)
n.batch <- 4
batch.length <- 25
n.burn <- 20
n.thin <- 1
n.chains <- 1

out <- spAbund(formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3 +
                           (1 | abund.factor.1) + (abund.cov.1 | abund.factor.2),
               data = data.list,
               n.batch = n.batch,
               batch.length = batch.length,
               inits = inits.list,
               tuning = tuning,
               priors = prior.list,
               NNGP = TRUE,
               cov.model = 'exponential',
               search.type = 'cb',
               n.neighbors = 5,
               accept.rate = 0.43,
               n.omp.threads = 1,
               verbose = TRUE,
               n.report = 1,
               n.burn = n.burn,
               n.thin = n.thin,
               n.chains = n.chains)
summary(out)
```

---

spDS *Function for Fitting Single-Species Spatially-Explicit Hierarchical Distance Sampling Models*

Description

Function for fitting single-sepcies spatially-explicit hierarchical distance sampling models. Spatial models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
spDS(abund.formula, det.formula, data, inits, priors, tuning,
     cov.model = 'exponential', NNGP = TRUE,
     n.neighbors = 15, search.type = 'cb',
     n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
     transect = 'line', det.func = 'halfnormal',
     n.omp.threads = 1, verbose = TRUE,
     n.report = 100, n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
     n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, dist.breaks, and offset. y is a matrix or data frame of the observed count values, with first dimension equal to the number of sites (J) and second dimension equal to the number of distance bins. covs is a matrix or data frame containing the variables used in the abundance and/or the detection portion of the model, with J rows for each column (variable). dist.breaks is a vector of distances that denote the breakpoints of the distance bands. dist.breaks should have length equal to the number of columns in y plus one. offset is an offset that can be used to scale estimates from abundance per transect to density per some desired unit of measure. This can be either a single value or a vector with an offset value for each site (e.g., if transects differ in length). coords is a $J \times 2$ matrix of the observation coordinates. Note that spAbundance assumes coordinates are specified in a projected coordinate system.
inits	a list with each tag corresponding to a parameter name. Valid tags are N, beta, alpha, kappa, sigma.sq, phi, w, nu, sigma.sq.mu, and sigma.sq.p. The value portion of each tag is the parameter's initial value. sigma.sq.mu and sigma.sq.p are only relevant when including random effects in the abundance and detection portion of the abundance model, respectively. kappa is only relevant when family = 'NB'. nu is only specified if cov.model = "matern". See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.normal, alpha.normal, kappa.unif, phi.unif, sigma.sq.ig, nu.unif, sigma.sq.mu.ig, and sigma.sq.p.ig. Abundance (beta) and detection (alpha) regression coefficients are assumed to follow a normal distribution. The hyperparameters of

the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances set to 100. The spatial variance parameter, `sigma.sq`, is assumed to follow an inverse-Gamma distribution. The spatial decay `phi`, spatial smoothness `nu`, and negative binomial dispersion `kappa` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for `sigma.sq` are passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts/slopes or of length one if priors are the same for all random effect variances.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), `kappa`, `phi`, `nu`, and `w`. See Roberts and Rosenthal (2009) for details.

`NNGP` if `TRUE`, model is fit with an NNGP. See Datta et al. (2016) and Finley et al. (2019) for more information. Currently only NNGP is supported, functionality for a Gaussian Process may be addded in future package development.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

accept.rate	target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.
family	the distribution to use for the latent abundance process. Currently supports 'NB' (negative binomial) and 'Poisson'.
transect	the type of transect. Currently supports line transects ('line') or circular transects (i.e., point counts; 'point').
det.func	the detection model used to describe how detection probability varies with distance. In other software, this is often referred to as the key function. Currently supports two functions: half normal ('halfnormal') and negative exponential ('negexp').
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hypterthreaded cores. Note, n.omp.threads > 1 might not work on some systems. Currently only relevant for spatial models.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report MCMC progress.
n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class spDS that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance dispersion parameter. Only included when family = 'NB'.
N.samples	a coda object of posterior samples for the latent abundance values. Note that these values always represent transect-level abundance, even when an offset is supplied.
mu.samples	a coda object of posterior samples for the latent expected abundance values. When an offset is supplied in the data object, these correspond to expected abundance per unit area (i.e., density).
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a coda object of posterior samples for latent spatial random effects.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.

`sigma.sq.p.samples`
: a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in `det.formula`.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `abund.formula`.

`alpha.star.samples`
: a coda object of posterior samples for the detection random effects. Only included if random effects are specified in `det.formula`.

`y.rep.samples`
: a three-dimensional array of fitted values. Array dimensions correspond to MCMC samples, sites, and distance band.

`pi.samples`
: a three-dimensional array of cell-specific detection probabilities. Array dimensions correspond to MCMC samples, sites, and distance band.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

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

Examples

```
set.seed(123)
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of distance bins from which to simulate data.
n.bins <- 5
```

```
# Length of each bin. This should be of length n.bins
bin.width <- c(.10, .10, .20, .3, .1)
# Abundance coefficients
beta <- c(1.0, 0.2, 0.3, -0.2)
p.abund <- length(beta)
# Detection coefficients
alpha <- c(-1.0, -0.3)
p.det <- length(alpha)
# Detection decay function
det.func <- 'halfnormal'
mu.RE <- list()
p.RE <- list()
sp <- TRUE
phi <- 3 / .5
sigma.sq <- 0.8
cov.model <- 'exponential'
family <- 'NB'
kappa <- 0.1
offset <- 1.8
transect <- 'point'

dat <- simDS(J.x = J.x, J.y = J.y, n.bins = n.bins, bin.width = bin.width,
             beta = beta, alpha = alpha, det.func = det.func, kappa = kappa,
             mu.RE = mu.RE, p.RE = p.RE, sp = sp,
             offset = offset, transect = transect, phi = phi, sigma.sq = sigma.sq,
             cov.model = cov.model)

y <- dat$y
X <- dat$X
X.re <- dat$X.re
X.p <- dat$X.p
X.p.re <- dat$X.p.re
dist.breaks <- dat$dist.breaks
coords <- dat$coords

covs <- cbind(X, X.p)
colnames(covs) <- c('int.abund', 'abund.cov.1', 'abund.cov.2', 'abund.cov.3',
                    'int.det', 'det.cov.1')

data.list <- list(y = y,
                  covs = covs,
                  dist.breaks = dist.breaks,
                  coords = coords,
                  offset = offset)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 10),
                   alpha.normal = list(mean = 0,
                                       var = 10),
                   kappa.unif = c(0, 100),
                   phi.unif = c(3 / 1, 3 / .1),
                   sigma.sq.ig = c(2, 1))
# Starting values
```

```
inits.list <- list(alpha = 0,
                   beta = 0,
                   kappa = 1,
                   phi = 3 / .5,
                   sigma.sq = 1)
# Tuning values
tuning <- list(beta = 0.1, alpha = 0.1, beta.star = 0.3, alpha.star = 0.1,
               kappa = 0.2, phi = 1, w = 1)

out <- spDS(abund.formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3,
            det.formula = ~ det.cov.1,
            data = data.list,
            n.batch = 10,
            batch.length = 25,
            inits = inits.list,
            family = 'NB',
            det.func = 'halfnormal',
            transect = 'point',
            cov.model = 'exponential',
            NNGP = TRUE,
    n.neighbors = 5,
            tuning = tuning,
            priors = prior.list,
            accept.rate = 0.43,
            n.omp.threads = 1,
            verbose = TRUE,
            n.report = 100,
            n.burn = 100,
            n.thin = 1,
            n.chains = 1)
summary(out)
```

---

spNMix *Function for Fitting Single-Species Spatial N-Mixture Models*

Description

The function spNMix fits single-species spatial N-mixture models. Spatial models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
spNMix(abund.formula, det.formula, data, inits, priors, tuning,
       cov.model = 'exponential', NNGP = TRUE,
       n.neighbors = 15, search.type = 'cb',
       n.batch, batch.length, accept.rate = 0.43, family = 'Poisson',
       n.omp.threads = 1, verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
       n.chains = 1, ...)
```

Arguments

abund.formula	a symbolic description of the model to be fit for the abundance portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and random slopes are allowed using lme4 syntax (Bates et al. 2015).
det.formula	a symbolic description of the model to be fit for the detection portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and random slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, abund.covs, det.covs, offset, and coords. y is the count data matrix or data frame with first dimension equal to the number of sites (J) and second dimension equal to the maximum number of replicates at a given site. abund.covs is a matrix or data frame containing the variables used in the abundance portion of the model, with J rows for each column (variable). det.covs is a list of variables included in the detection portion of the model. Each list element is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector of length J while observation-level covariates are specified as a matrix or data frame with the number of rows equal to J and number of columns equal to the maximum number of replicates at a given site. coords is a $J \times 2$ matrix of the observation coordinates. Note that spAbundance assumes coordinates are specified in a projected coordinate system. offset is an offset to use in the abundance model (e.g., an area offset). This can be either a single value or a vector with an offset for each site (e.g., if survey area differed in size).
inits	a list with each tag corresponding to a parameter name. Valid tags are N, beta, alpha, sigma.sq, phi, w, nu, kappa, sigma.sq.mu, sigma.sq.p. nu is only specified if cov.model = "matern", sigma.sq.p is only specified if there are random effects in det.formula, sigma.sq.mu is only specified if there are random effects in abund.formula, and kappa is only specified when family = 'NB'. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.normal, alpha.normal, phi.unif, sigma.sq.ig, nu.unif, kappa.unif, sigma.sq.mu.ig, and sigma.sq.p.ig. Abundance (beta) and detection (alpha) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances for abundance coefficients are set to 100 and for detection coefficients set to 2.72. The spatial variance parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution. The spatial decay phi, spatial smoothness nu, and negative binomial dispersion kappa parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for sigma.sq are passed as a vector of length

two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a vector of length two with the first and second elements corresponding to the lower and upper support, respectively. `sigma.sq.mu` and `sigma.sq.p` are the random effect variances for any abundance or detection random effects, respectively, and are assumed to follow an inverse-Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts/slopes or of length one if priors are the same for all random effect variances.

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `beta`, `alpha`, `beta.star` (the abundance random effect values), `alpha.star` (the detection random effect values), `kappa`, `phi`, `nu`, and `w`. See Roberts and Rosenthal (2009) for details.

`NNGP` if `TRUE`, model is fit with an NNGP. See Datta et al. (2016) and Finley et al. (2019) for more information. Currently only NNGP is supported, functionality for a Gaussian Process may be addded in future package development.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.batch` the number of MCMC batches in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch in each chain to run for the Adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for Adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for the latent abundance process. Currently supports `'NB'` (negative binomial) and `'Poisson'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of

	hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report Metropolis sampler acceptance and MCMC progress.
n.burn	the number of samples out of the total n.batch * batch.length samples in each chain to discard as burn-in. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of MCMC chains to run in sequence.
...	currently no additional arguments

Value

An object of class spNMix that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
kappa.samples	a coda object of posterior samples for the abundance dispersion parameter. Only included when family = 'NB'.
N.samples	a coda object of posterior samples for the latent abundance values
mu.samples	a coda object of posterior samples for the latent expected abundance values
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a coda object of posterior samples for latent spatial random effects.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random intercepts/slopes included in the abundance portion of the model. Only included if random effects are specified in abund.formula.
sigma.sq.p.samples	a coda object of posterior samples for variances of random effects included in the detection portion of the model. Only included if random effects are specified in det.formula.
beta.star.samples	a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in abund.formula.
alpha.star.samples	a coda object of posterior samples for the detection random effects. Only included if random effects are specified in det.formula.
rhat	a list of Gelman-Rubin diagnostic values for some of the model parameters.
ESS	a list of effective sample sizes for some of the model parameters.
run.time	execution time reported using proc.time().

The return object will include additional objects used for subsequent prediction and/or model fit evaluation. Note that detection probability values are not included in the model object, but can be extracted using fitted().

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

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

Examples

```
set.seed(350)
# Simulate Data ------------------------------------------------------------
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0.5, 1.5)
p.abund <- length(beta)
alpha <- c(0.5, 1.2, -0.5)
p.det <- length(alpha)
mu.RE <- list()
p.RE <- list()
phi <- runif(1, 3 / 1, 3 / .1)
sigma.sq <- runif(1, 0.2, 1.5)
kappa <- 0.5
sp <- TRUE
cov.model <- 'exponential'
dat <- simNMix(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
               kappa = kappa, mu.RE = mu.RE, p.RE = p.RE, sp = sp,
               phi = phi, sigma.sq = sigma.sq, cov.model = cov.model,
               family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re
X.p <- dat$X.p
X.p.re <- dat$X.p.re
coords <- dat$coords
```

```
abund.covs <- X
colnames(abund.covs) <- c('int', 'abund.cov.1')

det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])

data.list <- list(y = y,
                  abund.covs = abund.covs,
                  det.covs = det.covs,
                  coords = coords)

# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.abund),
                                      var = rep(100, p.abund)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)),
                   kappa.unif = c(0, 10))
# Starting values
inits.list <- list(alpha = alpha,
                   beta = beta,
                   kappa = kappa,
                   w = rep(0, J),
                   phi = 3 / 0.5,
                   sigma.sq = 1,
                   N = apply(y, 1, max, na.rm = TRUE))
# Tuning values
tuning.list <- list(phi = 0.5, kappa = 0.5, beta = 0.1, alpha = 0.1, w = 0.1)


n.batch <- 4
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- spNMix(abund.formula = ~ abund.cov.1,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              n.batch = n.batch,
              batch.length = batch.length,
              inits = inits.list,
              priors = prior.list,
              NNGP = TRUE,
              cov.model = 'spherical',
              n.neighbors = 10,
              accept.rate = 0.43,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 1,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = n.chains)
```

```
summary(out)
```

`summary.abund` *Methods for abund Object*

Description

Methods for extracting information from fitted univariate GLMMs (abund).

Usage

```
## S3 method for class 'abund'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'abund'
print(x, ...)
## S3 method for class 'abund'
plot(x, param, density = TRUE, ...)
```

Arguments

`object, x`	object of class abund.
`quantiles`	for summary, posterior distribution quantiles to compute.
`digits`	for summary, number of digits to report.
`param`	parameter name for which to generate a traceplot. Valid names are `"beta"`, `"beta.star"`, `"sigma.sq.mu"`, `"tau.sq"`.
`density`	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
`...`	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class abund, including methods to the generic functions `print`, `summary`, and `plot`.

Value

No return value, called to display summary information of a abund object.

summary.DS *Methods for DS Object*

Description

Methods for extracting information from fitted single-species hierarchiacl distance sampling (DS) models.

Usage

```
## S3 method for class 'DS'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'DS'
print(x, ...)
## S3 method for class 'DS'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class DS.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "alpha", "alpha.star", "sigma.sq.p".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class DS, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a DS object.

summary.lfMsAbund *Methods for lfMsAbund Object*

Description

Methods for extracting information from fitted latent factor multivariate abundance GLMMs (lfMsAbund).

Usage

```
## S3 method for class 'lfMsAbund'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'lfMsAbund'
print(x, ...)
## S3 method for class 'lfMsAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class lfMsAbund.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "beta.comm", "tau.sq.beta", "lambda".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class lfMsAbund, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a lfMsAbund object.

summary.lfMsDS *Methods for lfMsDS Object*

Description

Methods for extracting information from fitted latent factor multi-species hierarchical distance sampling (lfMsDS) model.

Usage

```
## S3 method for class 'lfMsDS'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'lfMsDS'
print(x, ...)
## S3 method for class 'lfMsDS'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class lfMsDS.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha", "lambda".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class lfMsDS, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a lfMsDS object.

summary.lfMsNMix *Methods for lfMsNMix Object*

Description

Methods for extracting information from fitted latent factor multi-species N-mixture (lfMsNMix) model.

Usage

```
## S3 method for class 'lfMsNMix'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'lfMsNMix'
print(x, ...)
## S3 method for class 'lfMsNMix'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class lfMsNMix.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha", "lambda".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class lfMsNMix, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a lfMsNMix object.

summary.msAbund *Methods for msAbund Object*

Description

Methods for extracting information from fitted multivariate abundance GLMMs (msAbund).

Usage

```
## S3 method for class 'msAbund'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'msAbund'
print(x, ...)
## S3 method for class 'msAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class msAbund.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "beta.comm", "tau.sq.beta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class msAbund, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a msAbund object.

summary.msDS *Methods for msDS Object*

Description

Methods for extracting information from fitted multi-species hierarchical distance sampling (msDS) model.

Usage

```
## S3 method for class 'msDS'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'msDS'
print(x, ...)
## S3 method for class 'msDS'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class msDS.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class msDS, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a msDS object.

summary.msNMix *Methods for msNMix Object*

Description

Methods for extracting information from fitted multi-species N-mixture (msNMix) model.

Usage

```
## S3 method for class 'msNMix'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'msNMix'
print(x, ...)
## S3 method for class 'msNMix'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class msNMix.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class msNMix, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a msNMix object.

summary.NMix *Methods for NMix Object*

Description

Methods for extracting information from fitted single-species N-mixture (NMix) model.

Usage

```
## S3 method for class 'NMix'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'NMix'
print(x, ...)
## S3 method for class 'NMix'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class NMix.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "alpha", "alpha.star", "sigma.sq.p".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class NMix, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a NMix object.

summary.sfMsAbund *Methods for sfMsAbund Object*

Description

Methods for extracting information from fitted spatial factor multivariate abundance GLMMs (sfMsAbund).

Usage

```
## S3 method for class 'sfMsAbund'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'sfMsAbund'
print(x, ...)
## S3 method for class 'sfMsAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class sfMsAbund.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "beta.comm", "tau.sq.beta", "lambda", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class sfMsAbund, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a sfMsAbund object.

summary.sfMsDS *Methods for sfMsDS Object*

Description

Methods for extracting information from fitted spatial multi-species hierarchical distance sampling (sfMsDS) model.

Usage

```
## S3 method for class 'sfMsDS'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'sfMsDS'
print(x, ...)
## S3 method for class 'sfMsDS'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class sfMsDS.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha", "lambda", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class sfMsDS, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a sfMsDS object.

summary.sfMsNMix *Methods for sfMsNMix Object*

Description

Methods for extracting information from fitted spatial factor multi-species N-mixture (sfMsNMix) model.

Usage

```
## S3 method for class 'sfMsNMix'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'sfMsNMix'
print(x, ...)
## S3 method for class 'sfMsNMix'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class sfMsNMix.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha", "lambda", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class sfMsNMix, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a sfMsNMix object.

summary.spAbund *Methods for spAbund Object*

Description

Methods for extracting information from fitted univariate spatially-explicit GLMMs (spAbund).

Usage

```
## S3 method for class 'spAbund'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spAbund'
print(x, ...)
## S3 method for class 'spAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spAbund.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class spAbund, including methods to the generic functions print, summary, plot.

Value

No return value, called to display summary information of a spAbund object.

summary.spDS *Methods for spDS Object*

Description

Methods for extracting information from fitted single-species spatial hierarchiacl distance sampling (spDS) models.

Usage

```
## S3 method for class 'spDS'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spDS'
print(x, ...)
## S3 method for class 'spDS'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spDS.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "alpha", "alpha.star", "sigma.sq.p", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class spDS, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a spDS object.

summary.spNMix *Methods for spNMix Object*

Description

Methods for extracting information from fitted single-species spatial N-mixture (spNMix) models.

Usage

```
## S3 method for class 'spNMix'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spNMix'
print(x, ...)
## S3 method for class 'spNMix'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spNMix.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "alpha", "alpha.star", "sigma.sq.p", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class spNMix, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a spNMix object.

summary.svcAbund *Methods for svcAbund Object*

Description

Methods for extracting information from fitted univariate spatially-varying coefficient GLMMs (svcAbund).

Usage

```
## S3 method for class 'svcAbund'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcAbund'
print(x, ...)
## S3 method for class 'svcAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcAbund.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class svcAbund, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcAbund object.

summary.svcMsAbund *Methods for svcMsAbund Object*

Description

Methods for extracting information from fitted multivariate spatially-varying coefficient abundance GLMMs (svcMsAbund).

Usage

```
## S3 method for class 'svcMsAbund'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcMsAbund'
print(x, ...)
## S3 method for class 'svcMsAbund'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcMsAbund.
level	a quoted keyword that indicates the level to summarize the model results. Valid key words are: "community", "species", or "both".
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
param	parameter name for which to generate a traceplot. Valid names are "beta", "beta.star", "sigma.sq.mu", "tau.sq", "beta.comm", "tau.sq.beta", "lambda", "theta".
density	logical value indicating whether to also generate a density plot for each parameter in addition to the MCMC traceplot.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class svcMsAbund, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcMsAbund object.

svcAbund *Function for Fitting Univariate Spatialy-Varying Coefficient GLMMs*

Description

The function svcAbund fits univariate spatially-varying coefficient GLMMs.

Usage

```
svcAbund(formula, data, inits, priors, tuning,
         svc.cols = 1, cov.model = 'exponential', NNGP = TRUE,
         n.neighbors = 15, search.type = 'cb', n.batch,
         batch.length, accept.rate = 0.43, family = 'Gaussian',
         n.omp.threads = 1, verbose = TRUE, n.report = 100,
         n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
         n.chains = 1, ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, and z (for family = 'zi-Gaussian' only. y is a vector of the observed count values, where the values represent the observed values at each site. covs is a list, matrix, or data frame of covariates used in the model, where each column (or list element) represents a different covariate. coords is a $J \times 2$ matrix of the observation coordinates. Note that svcAbundance assumes coordinates are specified in a projected coordinate system. z is used for fitting a zero-inflated Gaussian model. It is a vector where each value indicates the binary component of the model. In the context of abundance models, this can be thought of as the component of the model that indicates whether the species is present at each location, and then the supplied values in y are the observed abundance values at those locations where z = 1.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, phi, w, nu, tau.sq, sigma.sq.mu. nu is only specified if cov.model = "matern", sigma.sq.mu is only specified if there are random effects in formula, and The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name. Additionally, the tag fix can be set to TRUE to fix the starting values across all chains. If fix is not specified (the default), starting values are varied randomly across chains.
priors	a list with each tag corresponding to a parameter name. Valid tags are beta.normal, phi.unif, sigma.sq.ig, nu.unif, tau.sq.ig, sigma.sq.mu.ig. Abundance (beta) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the

normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances are set to 100. The spatial variance parameter, `sigma.sq`, and the Gaussian residual variance parameter, `tau.sq`, are assumed to follow an inverse-Gamma distribution. The spatial decay `phi` and spatial smoothness `nu`, parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for `sigma.sq` is passed as a list of length two with the first and second elements corresponding to the shape and scale parameters of the inverse-Gamma distribution either for each spatially-varying coefficient, or a single value if assumign the same values for all spatially-varying coefficients. The hyperparameters of the inverse-Gamma for `tau.sq` is passed as a vector of length two, with the first and second elements corresponding to the *shape* and *scale*, respectively. The hyperparameters of the Uniform are also passed as a list of length two with the first and second elements corresponding to the lower and upper support, respectively, for each SVC or a single value if giving the same prior for each SVC. `sigma.sq.mu` are the random effect variances for any random effects, and are assumed to follow an inverse-Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random effects or of length one if priors are the same for all random effect variances.

`svc.cols` a vector indicating the variables whose effects will be estimated as spatially-varying coefficients. `svc.cols` can be an integer vector with values indicating the order of covariates specified in the model formula (with 1 being the intercept if specified), or it can be specified as a character vector with names corresponding to variable names in `occ.covs` (for the intercept, use '(Intercept)'). `svc.cols` default argument of 1 results in a univariate spatial GLMM analogous to `spAbund` (assuming an intercept is included in the model).

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `phi` and `nu`. See Roberts and Rosenthal (2009) for details.

`NNGP` if `TRUE`, model is fit with an NNGP. See Datta et al. (2016) and Finley et al. (2019) for more information. Currently only NNGP is supported, functionality for a full GP may be addded in future package development.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the

	axis used for the nearest neighbor ordering then "cb" and "brute" should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then "cb" and "brute" might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.
n.batch	the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.
batch.length	the length of each MCMC batch in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.
accept.rate	target acceptance rate for adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.
family	the distribution to use for abundance. Currently, spatially-varying coefficient models are available for family = 'Gaussian' and family = 'zi-Gaussian'.
n.omp.threads	a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting n.omp.threads up to the number of hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report Metropolis sampler acceptance and MCMC progress.
n.burn	the number of samples out of the total n.batch * batch.length samples in each chain to discard as burn-in. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of MCMC chains to run in sequence.
...	currently no additional arguments

Value

An object of class svcAbund that is a list comprised of:

beta.samples	a coda object of posterior samples for the abundance regression coefficients.
tau.sq.samples	a coda object of posterior samples for the residual variance parameter.
y.rep.samples	a coda object of posterior samples for the abundance replicate (fitted) values with dimensions corresponding to MCMC samples and site.
mu.samples	a coda object of posterior samples for the expected abundance samples with dimensions corresponding to MCMC samples and site.
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a three-dimensional array of posterior samples for the spatially-varying coefficients with dimensions corresponding to MCMC sample, SVC, and site.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the model. Only included if random effects are specified in formula.

`beta.star.samples`
: a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in `formula`.

`like.samples`
: a coda object of posterior samples for the likelihood value associated with each site. Used for calculating WAIC.

`rhat`
: a list of Gelman-Rubin diagnostic values for some of the model parameters.

`ESS`
: a list of effective sample sizes for some of the model parameters.

`run.time`
: execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Examples

```
set.seed(1000)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Abundance ---------------------------
beta <- c(5, 0.5, -0.2, 0.75)
p <- length(beta)
mu.RE <- list()
mu.RE <- list(levels = c(35, 40),
              sigma.sq.mu = c(0.7, 1.5),
              beta.indx = list(1, 1))
# Spatial parameters ------------------
sp <- TRUE
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
cov.model <- "exponential"
```

```
sigma.sq <- runif(p.svc, 0.4, 4)
phi <- runif(p.svc, 3/1, 3/0.6)
tau.sq <- 2
z <- rbinom(J, 1, 0.5)

# Get all the data
dat <- simAbund(J.x = J.x, J.y = J.y, beta = beta, tau.sq = tau.sq,
                mu.RE = mu.RE, sp = sp, svc.cols = svc.cols,
                family = 'zi-Gaussian', cov.model = cov.model,
                sigma.sq = sigma.sq, phi = phi, z = z)
# Get data in format for spAbundance ------------------------------------
y <- dat$y
X <- dat$X
X.re <- dat$X.re
coords <- dat$coords

# Package all data into a list
covs <- cbind(X, X.re)
colnames(covs) <- c('int', 'cov.1', 'cov.2', 'cov.3', 'factor.1', 'factor.2')

# Data list bundle
data.list <- list(y = y, covs = covs, coords = coords, z = z)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 1000),
                   sigma.sq.ig = list(a = 2, b = 1), tau.sq = c(2, 1),
                   sigma.sq.mu.ig = list(a = 2, b = 1),
                   phi.unif = list(a = 3 / 1, b = 3 / 0.1))

# Starting values
inits.list <- list(beta = 0, alpha = 0,
                   sigma.sq = 1, phi = 3 / 0.5,
                   tau.sq = 2, sigma.sq.mu = 0.5)
# Tuning
tuning.list <- list(phi = 1)

n.batch <- 10
batch.length <- 25
n.burn <- 100
n.thin <- 1

out <- svcAbund(formula = ~ cov.1 + cov.2 + cov.3 +
                            (1 | factor.1) + (1 | factor.2),
                svc.cols = c(1, 2),
                data = data.list,
                n.batch = n.batch,
                batch.length = batch.length,
                inits = inits.list,
                priors = prior.list,
                accept.rate = 0.43,
                family = 'zi-Gaussian',
                cov.model = "exponential",
                tuning = tuning.list,
                n.omp.threads = 1,
```

```
verbose = TRUE,
NNGP = TRUE,
n.neighbors = 5,
n.report = 25,
n.burn = n.burn,
n.thin = n.thin,
n.chains = 3)
```

---

svcMsAbund *Function for Fitting Spatially-Varying Coefficient Multivariate Abundance GLMMs*

Description

The function svcMsAbund fits multivariate spatially-varying coefficient GLMs with species correlations (i.e., a spatially-explicit abundace-based joint species distribution model). We use a spatial factor modeling approach. Models are implemented using a Nearest Neighbor Gaussian Process.

Usage

```
svcMsAbund(formula, data, inits, priors, tuning,
           svc.cols = 1, cov.model = 'exponential', NNGP = TRUE,
           n.neighbors = 15, search.type = 'cb', n.factors,
           n.batch, batch.length, accept.rate = 0.43, family = 'Gaussian',
           n.omp.threads = 1, verbose = TRUE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1, n.chains = 1,
           ...)
```

Arguments

formula	a symbolic description of the model to be fit for the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts and slopes are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, and z. y is a matrix with sites corresponding to species and columns corresponding to sites. covs is a list, matrix, or data frame of covariates used in the model, where each column (or list element) represents a different covariate. coords is a $J \times 2$ matrix of the observation coordinates. Note that spAbundance assumes coordinates are specified in a projected coordinate system. For zero-inflated Gaussian models, the tag z is used to specify the binary component of the model and should have the same dimensions as y.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta.comm, beta, tau.sq.beta, sigma.sq.mu, phi, lambda, nu, and tau.sq. nu is only specified if cov.model = "matern", tau.sq is only specified for Gaussian and zero-inflated Gaussian models, and sigma.sq.mu is only specified if random effects are included in formula. The value portion of each tag is the parameter's initial value. See priors description for definition of each parameter name.

Additionally, the tag `fix` can be set to `TRUE` to fix the starting values across all chains. If `fix` is not specified (the default), starting values are varied randomly across chains.

`priors` a list with each tag corresponding to a parameter name. Valid tags are `beta.comm.normal`, `tau.sq.beta.ig`, `sigma.sq.mu`, `phi.unif`, `nu.unif`, and `tau.sq.ig`. Community-level (`beta.comm`) regression coefficients are assumed to follow a normal distribution. The hyperparameters of the normal distribution are passed as a list of length two with the first and second elements corresponding to the mean and variance of the normal distribution, which are each specified as vectors of length equal to the number of coefficients to be estimated or of length one if priors are the same for all coefficients. If not specified, prior means are set to 0 and prior variances to 100. Community-level variance parameters (`tau.sq.beta`) are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse gamma distribution are passed as a list of length two with the first and second elements corresponding to the shape and scale parameters, which are each specified as vectors of length equal to the number of coefficients to be estimated or a single value if priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. If desired, the species-specific regression coefficients (`beta`) can also be estimated indepdendently by specifying the tag `independent.betas = TRUE`. If specified, this will not estimate species-specific coefficients as random effects from a common-community-level distribution, and rather the values of `beta.comm` and `tau.sq.beta` will be fixed at the specified initial values. This is equivalent to specifying a Gaussian, independent prior for each of the species-specific effects. The spatial factor model fits `n.factors` independent spatial processes. The spatial decay `phi` and smoothness `nu` parameters for each latent factor and spatially-varying coefficient are assumed to follow Uniform distributions. The hyperparameters of the Uniform are passed as a list with two elements, with both elements being vectors of length equal to the number of spatial factors times the number of spatially-varying coefficients corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factors and spatially-varying coefficients. The priors for the factor loadings matrix `lambda` are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen and Amemlya 2002). The upper triangular elements of the `n.sp` x `n.factors` matrix for each spatially-varying coefficient are fixed at 0 and the diagonal elements are fixed at 1. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1). `sigma.sq.mu` are the random effect variances random effects, respectively, and are assumed to follow an inverse Gamma distribution. The hyperparameters of the inverse-Gamma distribution are passed as a list of length two with first and second elements corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of random intercepts or of length one if priors are the same for all random effect variances. `tau.sq` is the species-specific residual variance for Gaussian (or zero-inflated Gaussian) models, and it is assigned an inverse-Gamma prior. The hyperparameters of the inverse-Gamma are passed as a list of length two, with the first and second element corresponding to the shape and scale parameters, respectively, which are each specified as vectors of length equal to the number of species or a single value if priors are the same for

all species.

`tuning` a single numeric value representing the initial variance of the adaptive sampler for `phi` and `nu`. See Roberts and Rosenthal (2009) for details.

`svc.cols` a vector indicating the variables whose effects will be estimated as spatially-varying coefficients. `svc.cols` can be an integer vector with values indicating the order of covariates specified in the model formula (with 1 being the intercept if specified), or it can be specified as a character vector with names corresponding to variable names in `occ.covs` (for the intercept, use `'(Intercept)'`). `svc.cols` default argument of 1 results in a spatial factor model analogous to `sfMsAbund` (assuming an intercept is included in the model).

`cov.model` a quoted keyword that specifies the covariance function used to model the spatial dependence structure among the observations. Supported covariance model key words are: `"exponential"`, `"matern"`, `"spherical"`, and `"gaussian"`.

`NNGP` if `TRUE`, model is fit with an NNGP. If `FALSE`, a full Gaussian process is used. See Datta et al. (2016) and Finley et al. (2019) for more information. For spatial factor models, only `NNGP = TRUE` is currently supported.

`n.neighbors` number of neighbors used in the NNGP. Only used if `NNGP = TRUE`. Datta et al. (2016) showed that 15 neighbors is usually sufficient, but that as few as 5 neighbors can be adequate for certain data sets, which can lead to even greater decreases in run time. We recommend starting with 15 neighbors (the default) and if additional gains in computation time are desired, subsequently compare the results with a smaller number of neighbors using WAIC.

`search.type` a quoted keyword that specifies the type of nearest neighbor search algorithm. Supported method key words are: `"cb"` and `"brute"`. The `"cb"` should generally be much faster. If locations do not have identical coordinate values on the axis used for the nearest neighbor ordering then `"cb"` and `"brute"` should produce identical neighbor sets. However, if there are identical coordinate values on the axis used for nearest neighbor ordering, then `"cb"` and `"brute"` might produce different, but equally valid, neighbor sets, e.g., if data are on a grid.

`n.factors` the number of factors to use in the spatial factor model approach for each spatially-varying coefficient. Typically, the number of factors is set to be small (e.g., 4-5) relative to the total number of species in the community, which will lead to substantial decreases in computation time. However, the value can be anywhere between 1 and the number of species in the modeled community.

`n.batch` the number of MCMC batches in each chain to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`batch.length` the length of each MCMC batch to run for the adaptive MCMC sampler. See Roberts and Rosenthal (2009) for details.

`accept.rate` target acceptance rate for adaptive MCMC. Default is 0.43. See Roberts and Rosenthal (2009) for details.

`family` the distribution to use for abundance. Currently, spatially-varying coefficient models are available for `family = 'Gaussian'` and `family = 'zi-Gaussian'`.

`n.omp.threads` a positive integer indicating the number of threads to use for SMP parallel processing. The package must be compiled for OpenMP support. For most Intel-based machines, we recommend setting `n.omp.threads` up to the number of

	hyperthreaded cores. Note, n.omp.threads > 1 might not work on some systems.
verbose	if TRUE, messages about data preparation, model specification, and progress of the sampler are printed to the screen. Otherwise, no messages are printed.
n.report	the interval to report Metropolis sampler acceptance and MCMC progress. Note this is specified in terms of batches and not overall samples for spatial models.
n.burn	the number of samples out of the total n.samples to discard as burn-in for each chain. By default, the first 10% of samples is discarded.
n.thin	the thinning interval for collection of MCMC samples. The thinning occurs after the n.burn samples are discarded. Default value is set to 1.
n.chains	the number of chains to run in sequence.
...	currently no additional arguments

Value

An object of class svcMsAbund that is a list comprised of:

beta.comm.samples	a coda object of posterior samples for the community level regression coefficients.
tau.sq.beta.samples	a coda object of posterior samples for the abundance community variance parameters.
beta.samples	a coda object of posterior samples for the species level abundance regression coefficients.
tau.sq.samples	a coda object of posterior samples for the Gaussian residual variance parameter.
theta.samples	a coda object of posterior samples for the spatial correlation parameters.
lambda.samples	a coda object of posterior samples for the latent spatial factor loadings for each spatially-varying coefficient.
y.rep.samples	a three or four-dimensional array of posterior samples for the fitted (replicate) values for each species with dimensions corresponding to MCMC sample, species, site, and replicate.
mu.samples	a three or four-dimensional array of posterior samples for the expected abundance values for each species with dimensions corresponding to MCMC samples, species, site, and replicate.
w.samples	a four-dimensional array of posterior samples for the latent spatial random effects for each spatial factor within each spatially-varying coefficient. Dimensions correspond to MCMC sample, factor, site, and spatially-varying coefficient.
sigma.sq.mu.samples	a coda object of posterior samples for variances of random effects included in the abundance portion of the model. Only included if random effects are specified in abund.formula.
beta.star.samples	a coda object of posterior samples for the abundance random effects. Only included if random effects are specified in abund.formula.

`like.samples`	a three-dimensional array of posterior samples for the likelihood value associated with each site and species. Used for calculating WAIC.
`rhat`	a list of Gelman-Rubin diagnostic values for some of the model parameters.
`ESS`	a list of effective sample sizes for some of the model parameters.
`run.time`	MCMC sampler execution time reported using `proc.time()`.

The return object will include additional objects used for subsequent prediction and/or model fit evaluation.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@msu.edu>

References

Datta, A., S. Banerjee, A.O. Finley, and A.E. Gelfand. (2016) Hierarchical Nearest-Neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Roberts, G.O. and Rosenthal J.S. (2009) Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349-367.

Bates, Douglas, Martin Maechler, Ben Bolker, Steve Walker (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

Christensen, W. F., and Amemiya, Y. (2002). Latent variable analysis of multivariate spatial data. *Journal of the American Statistical Association*, 97(457), 302-317.

Examples

```
set.seed(332)
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- rep(1, J)
n.sp <- 6
# Community-level covariate effects
beta.mean <- c(0, 0.25, 0.6)
p.abund <- length(beta.mean)
tau.sq.beta <- c(0.2, 1.2, 0.4)
# Random effects
mu.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = n.sp, ncol = p.abund)
for (i in 1:p.abund) {
  beta[, i] <- rnorm(n.sp, beta.mean[i], sqrt(tau.sq.beta[i]))
}
sp <- TRUE
```

```
svc.cols <- c(1, 2)
n.factors <- 2
q.p.svc <- length(svc.cols) * n.factors
factor.model <- TRUE
phi <- runif(q.p.svc, 3/1, 3 / .4)
tau.sq <- runif(n.sp, 0.1, 5)
cov.model <- 'exponential'
family <- 'Gaussian'

dat <- simMsAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, n.sp = n.sp, beta = beta,
                  mu.RE = mu.RE, sp = sp, tau.sq = tau.sq, family = family,
                  factor.model = factor.model, phi = phi,
                  cov.model = cov.model, n.factors = n.factors,
                  svc.cols = svc.cols)

y <- dat$y
X <- dat$X
coords <- dat$coords

covs <- data.frame(abund.cov.1 = X[, 2],
                   abund.cov.2 = X[, 3])
data.list <- list(y = y, covs = covs, coords = coords)
prior.list <- list(beta.comm.normal = list(mean = 0, var = 100),
                   tau.sq.ig = list(a = 2, b = 2),
   phi.unif = list(a = 3 / 1, b = 3 / .1),
                   tau.sq.beta.ig = list(a = .1, b = .1))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   tau.sq = 1,
                   tau.sq.beta = 1,
                   phi = 3 / 0.5)
tuning.list <- list(phi = 0.5)

n.batch <- 5
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- svcMsAbund(formula = ~ abund.cov.1 + abund.cov.2,
                  data = data.list,
                  n.batch = n.batch,
                  inits = inits.list,
                  priors = prior.list,
                  tuning = tuning.list,
                  NNGP = TRUE,
                  svc.cols = c(1, 2),
                  family = 'Gaussian',
                  cov.model = 'exponential',
                  n.neighbors = 5,
                  n.factors = n.factors,
                  batch.length = batch.length,
                  n.omp.threads = 1,
```

```
                  verbose = TRUE,
                  n.report = 20,
                  n.burn = n.burn,
                  n.thin = n.thin,
                  n.chains = n.chains)
summary(out)
```

---

waicAbund *Compute Widely Applicable Information Criterion for spAbundance Model Objects*

Description

Function for computing the Widely Applicable Information Criterion (WAIC; Watanabe 2010) for spAbundance model objects.

Usage

```
waicAbund(object, N.max, by.species = FALSE, ...)
```

Arguments

object	an object of class NMix, spNMix, msNMix, lfMsNMix, sfMsNMix, abund, spAbund, msAbund, lfMsAbund, sfMsAbund, DS, spDS, msDS, lfMsDS, sfMsDS.
N.max	values indicating the upper limit on the latent abundance values when calculating WAIC for N-mixture models or hierarchical distance sampling models. For single-species models, this can be a single value or a vector of different values for each site. For multi-species models, this can be a single value, a vector of values for each species, or a species by site matrix for a separate value for each species/site combination. Defaults to ten plus the largest abundance value for each site/species in the posterior samples object$N.samples.
by.species	a logical value indicating whether or not WAIC should be reported individually for each species (TRUE) or summed across the entire community (FALSE) for multi-species models. Ignored for single species models.
...	currently no additional arguments

Details

The effective number of parameters is calculated following the recommendations of Gelman et al. (2014).

Value

Returns a vector with three elements corresponding to estimates of the expected log pointwise predictive density (elpd), the effective number of parameters (pD), and the WAIC. If calculating WAIC for a multi-species model and by.species = TRUE, this will be a data frame with rows corresponding to the different species.

Note

When fitting zero-inflated Gaussian models, the WAIC is only calculated for the non-zero values. If fitting a first stage model with spOccupancy to the binary portion of the zero-inflated model, you can use the spOccupancy::waicOcc function to calculate WAIC for the binary component.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,

References

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11:3571-3594.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. (2013). Bayesian Data Analysis. 3rd edition. CRC Press, Taylor and Francis Group

Gelman, A., J. Hwang, and A. Vehtari (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, 24:997-1016.

Examples

```
set.seed(1010)
J.x <- 15
J.y <- 15
J <- J.x * J.y
n.rep <- sample(3, J, replace = TRUE)
beta <- c(0, -1.5, 0.3, -0.8)
p.abund <- length(beta)
mu.RE <- list(levels = c(30), sigma.sq.mu = c(1.3))
kappa <- 0.5
sp <- FALSE
family <- 'NB'
dat <- simAbund(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta,
                kappa = kappa, mu.RE = mu.RE, sp = sp, family = 'NB')

y <- dat$y
X <- dat$X
X.re <- dat$X.re

abund.covs <- list(int = X[, , 1],
                   abund.cov.1 = X[, , 2],
                   abund.cov.2 = X[, , 3],
                   abund.cov.3 = X[, , 4],
                   abund.factor.1 = X.re[, , 1])

data.list <- list(y = y, covs = abund.covs)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 100),
```

```
                    kappa.unif = c(0.001, 10))
# Starting values
inits.list <- list(beta = 0, kappa = kappa)

n.batch <- 5
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.chains <- 1

out <- abund(formula = ~ abund.cov.1 + abund.cov.2 + abund.cov.3 +
                         (1 | abund.factor.1),
             data = data.list,
             n.batch = n.batch,
             batch.length = batch.length,
             inits = inits.list,
             priors = prior.list,
             accept.rate = 0.43,
             n.omp.threads = 1,
             verbose = TRUE,
             n.report = 1,
             n.burn = n.burn,
             n.thin = n.thin,
             n.chains = n.chains)

# Calculate WAIC
waicAbund(out)
```

Index

∗ **datasets**
bbsData, 8
bbsPredData, 9
dataNMixSim, 10
hbefCount2015, 29
neonDWP, 63
neonPredData, 64
∗ **model**
fitted.abund, 16
fitted.DS, 17
fitted.lfMsAbund, 18
fitted.lfMsDS, 18
fitted.lfMsNMix, 19
fitted.msAbund, 20
fitted.msDS, 21
fitted.msNMix, 21
fitted.NMix, 22
fitted.sfMsAbund, 23
fitted.sfMsDS, 24
fitted.sfMsNMix, 25
fitted.spAbund, 26
fitted.spDS, 26
fitted.spNMix, 27
fitted.svcAbund, 28
fitted.svcMsAbund, 29
summary.abund, 186
summary.DS, 187
summary.lfMsAbund, 188
summary.lfMsDS, 189
summary.lfMsNMix, 190
summary.msAbund, 191
summary.msDS, 192
summary.msNMix, 193
summary.NMix, 194
summary.sfMsAbund, 195
summary.sfMsDS, 196
summary.sfMsNMix, 197
summary.spAbund, 198
summary.spDS, 199
summary.spNMix, 200
summary.svcAbund, 201
summary.svcMsAbund, 202

abund, 3

bbsData, 8
bbsPredData, 9

dataNMixSim, 10
DS, 11

fitted, *17–29*
fitted.abund, 16
fitted.DS, 17
fitted.lfMsAbund, 18
fitted.lfMsDS, 18
fitted.lfMsNMix, 19
fitted.msAbund, 20
fitted.msDS, 21
fitted.msNMix, 21
fitted.NMix, 22
fitted.sfMsAbund, 23
fitted.sfMsDS, 24
fitted.sfMsNMix, 25
fitted.spAbund, 26
fitted.spDS, 26
fitted.spNMix, 27
fitted.svcAbund, 28
fitted.svcMsAbund, 29

hbefCount2015, 29

lfMsAbund, 30
lfMsDS, 35
lfMsNMix, 42

msAbund, 47
msDS, 52
msNMix, 58

neonDWP, 63
neonPredData, 64
NMix, 65

plot, *186–202*
plot.abund *(*summary.abund*)*, 186
plot.DS *(*summary.DS*)*, 187
plot.lfMsAbund *(*summary.lfMsAbund*)*, 188
plot.lfMsDS *(*summary.lfMsDS*)*, 189
plot.lfMsNMix *(*summary.lfMsNMix*)*, 190
plot.msAbund *(*summary.msAbund*)*, 191
plot.msDS *(*summary.msDS*)*, 192
plot.msNMix *(*summary.msNMix*)*, 193
plot.NMix *(*summary.NMix*)*, 194
plot.sfMsAbund *(*summary.sfMsAbund*)*, 195
plot.sfMsDS *(*summary.sfMsDS*)*, 196
plot.sfMsNMix *(*summary.sfMsNMix*)*, 197
plot.spAbund *(*summary.spAbund*)*, 198
plot.spDS *(*summary.spDS*)*, 199
plot.spNMix *(*summary.spNMix*)*, 200
plot.svcAbund *(*summary.svcAbund*)*, 201
plot.svcMsAbund *(*summary.svcMsAbund*)*, 202
ppcAbund, 69
predict.abund, 72
predict.DS, 75
predict.lfMsAbund, 78
predict.lfMsDS, 81
predict.lfMsNMix, 85
predict.msAbund, 89
predict.msDS, 91
predict.msNMix, 95
predict.NMix, 98
predict.sfMsAbund, 101
predict.sfMsDS, 104
predict.sfMsNMix, 109
predict.spAbund, 112
predict.spDS, 116
predict.spNMix, 120
predict.svcAbund, 123
predict.svcMsAbund, 127
print, *186–202*
print.abund *(*summary.abund*)*, 186
print.DS *(*summary.DS*)*, 187
print.lfMsAbund *(*summary.lfMsAbund*)*, 188
print.lfMsDS *(*summary.lfMsDS*)*, 189
print.lfMsNMix *(*summary.lfMsNMix*)*, 190
print.msAbund *(*summary.msAbund*)*, 191
print.msDS *(*summary.msDS*)*, 192
print.msNMix *(*summary.msNMix*)*, 193
print.NMix *(*summary.NMix*)*, 194
print.sfMsAbund *(*summary.sfMsAbund*)*, 195
print.sfMsDS *(*summary.sfMsDS*)*, 196
print.sfMsNMix *(*summary.sfMsNMix*)*, 197
print.spAbund *(*summary.spAbund*)*, 198
print.spDS *(*summary.spDS*)*, 199
print.spNMix *(*summary.spNMix*)*, 200
print.svcAbund *(*summary.svcAbund*)*, 201
print.svcMsAbund *(*summary.svcMsAbund*)*, 202

sfMsAbund, 130
sfMsDS, 136
sfMsNMix, 144
simAbund, 150
simDS, 153
simMsAbund, 156
simMsDS, 159
simMsNMix, 162
simNMix, 166
spAbund, 169
spDS, 174
spNMix, 180
summary, *186–202*
summary.abund, 186
summary.DS, 187
summary.lfMsAbund, 188
summary.lfMsDS, 189
summary.lfMsNMix, 190
summary.msAbund, 191
summary.msDS, 192
summary.msNMix, 193
summary.NMix, 194
summary.sfMsAbund, 195
summary.sfMsDS, 196
summary.sfMsNMix, 197
summary.spAbund, 198
summary.spDS, 199
summary.spNMix, 200
summary.svcAbund, 201
summary.svcMsAbund, 202
svcAbund, 203
svcMsAbund, 208

waicAbund, 214