Package 'spOccupancy'

November 1, 2023

Type Package

Title Single-Species, Multi-Species, and Integrated Spatial Occupancy
Models

Version 0.7.2

Author Jeffrey Doser [aut, cre],
Andrew Finley [aut],
Marc Kery [ctb]

Maintainer Jeffrey Doser <doserjef@msu.edu>

Description Fits single-species, multi-species, and integrated non-spatial and spatial occupancy models using Markov Chain Monte Carlo (MCMC). Models are fit using Polya-Gamma data augmentation detailed in Polson, Scott, and Windle (2013) <doi:10.1080/01621459.2013.829001>. Spatial models are fit using either Gaussian processes or Nearest Neighbor Gaussian Processes (NNGP) for large spatial datasets. 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>. Provides functionality for data integration of multiple single-species occupancy data sets using a joint likelihood framework. Details on data integration are given in Miller, Pacifici, Sanderlin, and Reich (2019) <doi:10.1111/2041-210X.13110>. Details on single-species and multi-species models are found in MacKenzie, Nichols, Lachman, Droege, Royle, and Langtimm (2002) <doi:10.1890/0012-9658(2002)083[2248:ESORWD]2.0.CO;2> and Dorazio and Royle <doi:10.1198/016214505000000015>, respectively.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://www.jeffdoser.com/files/spoccupancy-web,
https://github.com/doserjef/spOccupancy

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

Depends R (>= 3.5.0)

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

Suggests testthat

R **topics documented:**

spOccupancy–package	*Single-Species, Multi-Species, and Integrated Spatial Occupancy Models*

Description

Fits single-species, multi-species, and integrated non-spatial and spatial occupancy models using Markov Chain Monte Carlo (MCMC). Models are fit using Polya-Gamma data augmentation detailed in Polson, Scott, and Windle (2013). Spatial models are fit using either Gaussian processes or Nearest Neighbor Gaussian Processes (NNGP) for large spatial datasets. Details on NNGPs are given in Datta, Banerjee, Finley, and Gelfand (2016). Provides functionality for data integration of multiple occupancy data sets using a joint likelihood framework. Details on data integration are given in Miller, Pacifici, Sanderlin, and Reich (2019). Details on single-species and multi-species models are found in MacKenzie et al. (2002) and Dorazio and Royle (2005), respectively. Details on the package functionality is given in Doser et al. (2022), Doser, Finley, Banerjee (2023), and Doser, Finley, Saunders, Kery, Weed, and Zipkin (2023). See `citation('spOccupancy')` for how to cite spOccupancy in publications.

Single-species models

[PGOcc](#) fits single-species occupancy models.

[spPGOcc](#) fits single-species spatial occupancy models.

[intPGOcc](#) fits single-species integrated occupancy models (i.e., an occupancy model with multiple data sources).

[spIntPGOcc](#) fits single-species integrated spatial occupancy models.

[tPGOcc](#) fits a multi-season single-species occupancy model.

[stPGOcc](#) fits a multi-season single-species spatial occupancy model.

[svcPGBinom](#) fits a single-species spatially-varying coefficient GLM.

[svcPGOcc](#) fits a single-species spatially-varying coefficient occupancy model.

[svcTPGBinom](#) fits a single-species spatially-varying coefficient multi-season GLM.

svcTPGOcc fits a single-species spatially-varying coefficient multi-season occupancy model.

Multi-species models

msPGOcc fits multi-species occupancy models.

spMsPGOcc fits multi-species spatial occupancy models.

lfJSDM fits a joint species distribution model without imperfect detection.

sfJSDM fits a spatial joint species distribution model without imperfect detection.

lfMsPGOcc fits a joint species distribution model with imperfect detection (i.e., a multi-species occupancy model with residual species correlations).

sfMsPGOcc fits a spatial joint species distribution model with imperfect detection.

svcMsPGOcc fits a multi-species spatially-varying coefficient occupancy model.

tMsPGOcc fits a multi-season multi-species occupancy model.

stMsPGOcc fits a multi-season multi-species spatial occupancy model.

svcTMsPGOcc fits a multi-season multi-species spatially-varying coefficient occupancy model.

Goodness of Fit and Model Assessment Functions

ppcOcc performs posterior predictive checks.

waicOcc computes the Widely Applicable Information Criterion for spOccupancy model objects.

Data Simulation Functions

simOcc simulates single-species occupancy data.

simTOcc simulates single-species multi-season occupancy data.

simBinom simulates detection-nondetection data with perfect detection.

simTBinom simulates multi-season detection-nondetection data with perfect detection.

simMsOcc simulates multi-species occupancy data.

simIntOcc simulates single-species occupancy data from multiple data sources.

simTMsOcc simulates multi-species multi-season occupancy data from multiple data sources.

All objects from model-fitting functions have support with the summary function for displaying a concise summary of model results, the fitted function for extracting model fitted values, and the predict function for predicting occupancy and/or detection across an area of interest.

Author(s)

Jeffrey W. Doser, Andrew O. Finley, Marc Kery

References

Doser, J. W., Finley, A. O., Kery, M., & Zipkin, E. F. (2022). spOccupancy: An R package for single-species, multi-species, and integrated spatial occupancy models. Methods in Ecology and Evolution, 13, 1670-1678. doi:10.1111/2041210X.13897

Doser, J. W., Finley, A. O., & Banerjee, S. (2023). Joint species distribution models with imperfect detection for high-dimensional spatial data. Ecology e4137. doi:10.1002/ecy.4137

Doser, J. W., Finley, A. O., Saunders, S. P., Kery, M., Weed A. S., Zipkin, E. F. (2023). Modeling complex species-environment relationships through spatially-varying coefficient occupancy models. arXiv preprint.

---

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

Description

Method for extracting model fitted values and detection probability values from a fitted single-species integrated occupancy (intPGOcc) model.

Usage

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

Arguments

object   object of class intPGOcc.

...   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 intPGOcc.

Value

A list comprised of

y.rep.samples A list of three-dimensional numeric arrays of fitted values for each individual data source for use in Goodness of Fit assessments.

p.samples   A list of three-dimensional numeric arrays of detection probability values.

---

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

Description

Method for extracting model fitted values and probability values from a fitted latent factor joint species distribution model (lfJSDM).

Usage

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

Arguments

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

Details

A method to the generic [fitted](#) function to extract fitted values and probability values for fitted model objects of class `lfJSDM`.

Value

A list comprised of:

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

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

Description

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

Usage

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

Arguments

`object`	object of class `lfMsPGOcc`.
`...`	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 `lfMsPGOcc`.

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.msPGOcc`	*Extract Model Fitted Values for msPGOcc Object*

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species occupancy (`msPGOcc`) model.

Usage

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

Arguments

`object`	object of class `msPGOcc`.
`...`	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 `msPGOcc`.

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.PGOcc *Extract Model Fitted Values for PGOcc Object*

Description

Method for extracting model fitted values and detection probabilities from a fitted single-species occupancy (PGOcc) model.

Usage

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

Arguments

object   object of class PGOcc.

...   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 PGOcc.

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.sfJSDM *Extract Model Fitted Values for sfJSDM Object*

Description

Method for extracting model fitted values and probability values from a fitted spatial factor joint species distribution model (sfJSDM).

Usage

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

Arguments

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

Details

A method to the generic [`fitted`](#) function to extract fitted values and probability values for fitted model objects of class `sfJSDM`.

Value

A list comprised of:

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

---

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

Description

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

Usage

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

Arguments

`object`	object of class `sfMsPGOcc`.
`...`	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 `sfMsPGOcc`.

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.spIntPGOcc   *Extract Model Fitted Values for spIntPGOcc Object*

Description

Method for extracting model fitted values and detection probability values from a fitted single-species integrated spatial occupancy (`spIntPGOcc`) model.

Usage

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

Arguments

object   object of class `spIntPGOcc`.

...   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 `spIntPGOcc`.

Value

A list comprised of

y.rep.samples   A list of three-dimensional numeric arrays of fitted values for each individual data source for use in Goodness of Fit assessments.

p.samples   A list of three-dimensional numeric arrays of detection probability values.

---

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

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species spatial occupancy (spMsPGOcc) model.

Usage

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

Arguments

object	object of class spMsPGOcc.
...	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 spMsPGOcc.

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.spPGOcc   *Extract Model Fitted Values for spPGOcc Object*

Description

Method for extracting model fitted values and detection probabilities from a fitted single-species spatial occupancy (spPGOcc) model.

Usage

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

Arguments

`object`	object of class `spPGOcc`.
`...`	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 `spPGOcc`.

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.stMsPGOcc   *Extract Model Fitted Values for stMsPGOcc Object*

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species multi-season spatial occupancy (`stMsPGOcc`) model.

Usage

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

Arguments

`object`	object of class `stMsPGOcc`.
`...`	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 `stMsPGOcc`.

Value

A list comprised of:

y.rep.samples   A five-dimensional numeric array of fitted values for use in Goodness of Fit
                assessments. Array dimensions correspond to MCMC samples, species, sites,
                primary time period, and replicates.

p.samples   A five-dimensional numeric array of detection probability values. Array dimen-
                sions correspond to MCMC samples, species, sites, primary time period, and
                replicates.

---

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

Description

Method for extracting model fitted values and detection probabilities from a fitted multi-season
single-species spatial occupancy (stPGOcc) model.

Usage

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

Arguments

object   object of class stPGOcc.

...   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 stPGOcc.

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, sites, primary
                time periods, and replicates.

p.samples   A four-dimensional numeric array of detection probability values. Array dimen-
                sions correspond to MCMC samples, sites, primary time periods, and replicates.

---

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

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species spatially varying coefficient occupancy (svcMsPGOcc) model.

Usage

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

Arguments

object   object of class svcMsPGOcc.

...   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 svcMsPGOcc.

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.svcPGBinom   *Extract Model Fitted Values for svcPGBinom Object*

Description

Method for extracting model fitted values from a fitted single-species spatially-varying coefficients binomial model (svcPGBinom).

Usage

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

Arguments

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

Details

A method to the generic [fitted](#) function to extract fitted values for fitted model objects of class svcPGBinom.

Value

A two-dimensional matrix of fitted values for use in Goodness of Fit assessments. Dimensions correspond to MCMC samples and sites.

---

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

Description

Method for extracting model fitted values and detection probabilities from a fitted single-species spatially-varying coefficients occupancy (svcPGOcc) model.

Usage

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

Arguments

object	object of class svcPGOcc.
...	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 svcPGOcc.

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.svcTMsPGOcc   *Extract Model Fitted Values for svcTMsPGOcc Object*

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species multi-season spatially varying coefficient occupancy (svcTMsPGOcc) model.

Usage

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

Arguments

object	object of class svcTMsPGOcc.
...	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 svcTMsPGOcc.

Value

A list comprised of:

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

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

Description

Method for extracting model fitted values from a fitted multi-season single-species spatially-varying coefficients binomial model (svcTPGBinom).

Usage

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

Arguments

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

Details

A method to the generic [`fitted`](#) function to extract fitted values for fitted model objects of class `svcTPGBinom`.

Value

A three-dimensional matrix of fitted values for use in Goodness of Fit assessments. Dimensions correspond to MCMC samples, sites, and primary time periods.

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

Description

Method for extracting model fitted values and detection probabilities from a fitted multi-season single-species spatially-varying coefficients occupancy (`svcTPGOcc`) model.

Usage

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

Arguments

`object`	object of class `svcTPGOcc`.
`...`	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 `svcTPGOcc`.

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, sites, primary time periods, and replicates.
`p.samples`	A four-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, sites, primary time periods, and replicates.

---

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

Description

Method for extracting model fitted values and detection probability values from a fitted multi-species multi-season occupancy (`tMsPGOcc`) model.

Usage

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

Arguments

object	object of class `tMsPGOcc`.
...	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 `tMsPGOcc`.

Value

A list comprised of:

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

---

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

Description

Method for extracting model fitted values and detection probabilities from a fitted multi-season single-species occupancy (`tPGOcc`) model.

Usage

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

Arguments

`object`	object of class `tPGOcc`.
`...`	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 `tPGOcc`.

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, sites, primary time periods, and replicates.
`p.samples`	A four-dimensional numeric array of detection probability values. Array dimensions correspond to MCMC samples, sites, primary time periods, and replicates.

---

getSVCSamples   *Extract spatially-varying coefficient MCMC samples*

Description

Function for extracting the full spatially-varying coefficient MCMC samples from an spOccupancy model object.

Usage

```
getSVCSamples(object, pred.object, ...)
```

Arguments

`object`	an object of class `svcPGOcc`, `svcPGBinom`, `svcTPGOcc`, `svcTPGBinom`, `svcMsPGOcc`, `svcTMsPGOcc`.
`pred.object`	a prediction object from a spatially-varying coefficient model fit using spOccupancy. Should be of class `predict.svcPGOcc`, `predict.svcPGBinom`, `predict.svcTPGOcc`, `predict.svcTPGBinom`, `predict.svcMsPGOcc`, or `predict.svcTMsPGOcc`. If specified, SVC samples are extracted at the prediction locations.
`...`	currently no additional arguments

Value

A list of coda::mcmc objects of the spatially-varying coefficient MCMC samples for all spatially-varying coefficients estimated in the model (including the intercept if specified). Note these values correspond to the sum of the estimated spatial and non-spatial effect to give the overall effect of the covariate at each location. Each element of the list is a two-dimensional matrix where dimensions correspond to MCMC sample and site. If pred.object is specified, values are returned for the prediction locations instead of the sampled locations.

Note

For multi-species models, the value of the SVC will be returned at all spatial locations for each species even when range.ind is specified in the data list when fitting the model. This may not be desirable for complete summaries of the SVC for each species, so if specifying range.ind in the data list, you may want to subsequently process the SVC samples for each species to be restricted to each species range.

Author(s)

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

Examples

```
set.seed(400)
# Simulate Data -----------------------------------------------------------
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, 2)
p.occ <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
phi <- c(3 / .6, 3 / .8)
sigma.sq <- c(1.2, 0.7)
svc.cols <- c(1, 2)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sigma.sq = sigma.sq, phi = phi, sp = TRUE, cov.model = 'exponential',
              svc.cols = svc.cols)
# Detection-nondetection data
y <- dat$y
# Occupancy covariates
X <- dat$X
# Detection covarites
X.p <- dat$X.p
# Spatial coordinates
coords <- dat$coords

# Package all data into a list
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
```

```
det.covs <- list(det.cov.1 = X.p[, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = list(a = 2, b = 1),
                   phi.unif = list(a = 3/1, b = 3/.1))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
                   phi = 3 / .5,
                   sigma.sq = 2,
                   w = matrix(0, nrow = length(svc.cols), ncol = nrow(X)),
                   z = apply(y, 1, max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- svcPGOcc(occ.formula = ~ occ.cov,
                det.formula = ~ det.cov.1,
                data = data.list,
                inits = inits.list,
                n.batch = n.batch,
                batch.length = batch.length,
                accept.rate = 0.43,
                priors = prior.list,
                cov.model = 'exponential',
                svc.cols = c(1, 2),
                tuning = tuning.list,
                n.omp.threads = 1,
                verbose = TRUE,
                NNGP = TRUE,
                n.neighbors = 5,
                search.type = 'cb',
                n.report = 10,
                n.burn = 50,
                n.thin = 1)

svc.samples <- getSVCSamples(out)
str(svc.samples)
```

---

hbef2015   *Detection-nondetection data of 12 foliage gleaning bird species in*
                            2015 in the Hubbard Brook Experimental Forest

---

Description

Detection-nondetection 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(hbef2015)
```

Format

`hbef2015` is a list with four elements:

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

`occ.covs`: a numeric matrix 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

hbefElev	*Elevation in meters extracted at a 30m resolution across the Hubbard Brook Experimental Forest*

Description

Elevation in meters extracted at a 30m resolution of the Hubbard Brook Experimental Forest. Data come from the National Elevation Dataset.

Usage

```
data(hbefElev)
```

Format

hbefElev is a data frame with three columns:

val: the elevation value in meters.

Easting: the x coordinate of the point. The proj4string is "+proj=utm +zone=19 +units=m +datum=NAD83".

Northing: the y coordinate of the point. The proj4string is "+proj=utm +zone=19 +units=m +datum=NAD83".

Source

Gesch, D., Oimoen, M., Greenlee, S., Nelson, C., Steuck, M., & Tyler, D. (2002). The national elevation dataset. Photogrammetric engineering and remote sensing, 68(1), 5-32.

References

Gesch, D., Oimoen, M., Greenlee, S., Nelson, C., Steuck, M., & Tyler, D. (2002). The national elevation dataset. Photogrammetric engineering and remote sensing, 68(1), 5-32.

hbefTrends	*Detection-nondetection data of 12 foliage gleaning bird species from 2010-2018 in the Hubbard Brook Experimental Forest*

Description

Detection-nondetection data of 12 foliage gleaning bird species in 2010-2018 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(hbefTrends)
```

Format

hbefTrends is a list with four elements:

y: a four-dimensional array of detection-nondetection data with dimensions of species (12), sites (373), years (9), and replicates (3).

occ.covs: a list of potential covariates for inclusion in the occurrence portion of an occupancy model. There are two covariates: elevation (a site-level covariate), and years (a temporal covariate.) det.covs: a list of two numeric three-dimensional arrays with dimensions corresponding to sites (373), years (9), and replicates (3). The first element is the day of year when the survey was conducted for a given site, year, 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

---

intMsPGOcc — *Function for Fitting Integrated Multi-Species Occupancy Models Using Polya-Gamma Latent Variables*

Description

Function for fitting multi-species occupancy models using Polya-Gamma latent variables.

Usage

```
intMsPGOcc(occ.formula, det.formula, data, inits, priors, n.samples,
           n.omp.threads = 1, verbose = TRUE, n.report = 100,
           n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
           k.fold, k.fold.threads = 1, k.fold.seed, k.fold.only = FALSE, ...)
```

Arguments

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

det.formula   a list of symbolic descriptions of the models to be fit for the detection portion of the model using R's model syntax for each data set. Each element in the list is a formula for the detection model of a given data set. Only right-hand side of formula is specified. Random effects are not currently supported. See example below.

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, sites, and species. y is a list of three-dimensional arrays. Each element of the list has first dimension equal to the number of species observed in that data source, second dimension equal to the number of sites observed in that data source, and thir dimension equal to the maximum number of replicates at a given site. occ.covs is a matrix or data frame containing the variables used in the occurrence portion of the model, with the number of rows being the number of sites with at least one data source for each column (variable). det.covs is a list of variables included in the detection portion of the model for each data source. det.covs should have the same number of elements as y, where each element is itself a list. Each element of the list for a given data source is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector with length equal to the number of observed sites of that data source, while observational-level covariates are specified as a matrix or data frame with the number of rows equal to the number of observed sites of that data source and number of columns equal to the maximum number of replicates at a given site. sites is a list of site indices with number of elements equal to the number of data sources being modeled. Each element contains a vector of length equal to the number of sites that specific data source contains. Each value in the vector indicates the row in occ.covs that corresponds with the specific row of the detection-nondetection data for the data source. This is used to properly link sites across data sets. species is a list with number of data sources being modeled. Each element of the list is a vector of codes (these can be numeric or character) that indicate the species modeled in the specific data set.

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.psi, and z. 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.psi.ig, and sigma.sq.p.ig. Community-level occurrence (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 set to 2.72. For the community-level detection means (`alpha.comm`), the mean and variance hyperparameters are themselves passed in as lists, with each element of the list corresponding to the specific hyperparameters for the detection parameters in a given data source. If not specified, prior means are set to 0 and prior variances set to 2.72. Community-level variance parameters for occurrence (`tau.sq.beta`) and detection (`tau.sq.alpha`) are assumed to follow an inverse Gamma distribution. For the occurrence parameters, 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. For the detection community-level variance parameters (`tau.sq.alpha`), the shape and scale parameters are passed in as lists, with each element of the list corresponding to the specific hyperparameters for the detection variances in a given data source. `sigma.sq.psi` and are the random effect variances for any occurrence 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.

n.samples	the number of posterior samples to collect in each chain.
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.
k.fold	cross-validation is not currently supported for integrated multi-species occupancy models.
k.fold.threads	cross-validation is not currently supported for integrated multi-species occupancy models.
k.fold.seed	cross-validation is not currently supported for integrated multi-species occupancy models.
k.fold.only	cross-validation is not currently supported for integrated multi-species occupancy models.

...	currently no additional arguments

Value

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

beta.comm.samples

a coda object of posterior samples for the community level occurrence regression coefficients.

alpha.comm.samples

a coda object of posterior samples for the community level detection regression coefficients for all data sources.

tau.sq.beta.samples

a coda object of posterior samples for the occurrence community variance parameters.

tau.sq.alpha.samples

a coda object of posterior samples for the detection community variance parameters for all data sources.

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

alpha.samples   a coda object of posterior samples for the species level detection regression coefficients for all data sources.

z.samples   a three-dimensional array of posterior samples for the latent occurrence values for each species.

psi.samples   a three-dimensional array of posterior samples for the latent occurrence probability values for each species.

sigma.sq.psi.samples

a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

beta.star.samples

a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.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.

Note

Basic functionality of this function is stable, but some components are still in development and not currently available. Please create a GitHub issue on the package GitHub page if you use this function and encounter an error.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Dorazio, R. M., and Royle, J. A. (2005). Estimating size and composition of biological communities by modeling the occurrence of species. Journal of the American Statistical Association, 100(470), 389-398.

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

Examples

```
set.seed(91)
J.x <- 10
J.y <- 10
# Total number of data sources across the study region
J.all <- J.x * J.y
# Number of data sources.
n.data <- 2
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
n.rep <- list()
n.rep[[1]] <- rep(3, J.obs[1])
n.rep[[2]] <- rep(4, J.obs[2])

# Number of species observed in each data source
N <- c(8, 3)

# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.4, 0.3)
# Detection
# Detection covariates
alpha.mean <- list()
tau.sq.alpha <- list()
# Number of detection parameters in each data source
p.det.long <- c(4, 3)
for (i in 1:n.data) {
  alpha.mean[[i]] <- runif(p.det.long[i], -1, 1)
  tau.sq.alpha[[i]] <- runif(p.det.long[i], 0.1, 1)
}
```

```
# Random effects
psi.RE <- list()
p.RE <- list()
beta <- matrix(NA, nrow = max(N), ncol = p.occ)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(max(N), beta.mean[i], sqrt(tau.sq.beta[i]))
}
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- matrix(NA, nrow = N[i], ncol = p.det.long[i])
  for (t in 1:p.det.long[i]) {
    alpha[[i]][, t] <- rnorm(N[i], alpha.mean[[i]][t], sqrt(tau.sq.alpha[[i]])[t])
  }
}
sp <- FALSE
factor.model <- FALSE
# Simulate occupancy data
dat <- simIntMsOcc(n.data = n.data, J.x = J.x, J.y = J.y,
                   J.obs = J.obs, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                   psi.RE = psi.RE, p.RE = p.RE, sp = sp, factor.model = factor.model,
                   n.factors = n.factors)
J <- nrow(dat$coords.obs)
y <- dat$y
X <- dat$X.obs
X.p <- dat$X.p
X.re <- dat$X.re.obs
X.p.re <- dat$X.p.re
sites <- dat$sites
species <- dat$species

# Package all data into a list
occ.covs <- cbind(X)
colnames(occ.covs) <- c('int', 'occ.cov.1')
#colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2],
                      det.cov.1.2 = X.p[[1]][, , 3],
                      det.cov.1.3 = X.p[[1]][, , 4])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2],
                      det.cov.2.2 = X.p[[2]][, , 3])

data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites,
                  species = species)
# Take a look at the data.list structure for integrated multi-species
# occupancy models.
# Priors
prior.list <- list(beta.comm.normal = list(mean = 0,var = 2.73),
                   alpha.comm.normal = list(mean = list(0, 0),
                                                  var = list(2.72, 2.72)),
```

```
                            tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                            tau.sq.alpha.ig = list(a = list(0.1, 0.1),
                                                   b = list(0.1, 0.1)))
    inits.list <- list(alpha.comm = list(0, 0),
                       beta.comm = 0,
                       tau.sq.beta = 1,
                       tau.sq.alpha = list(1, 1),
                   alpha = list(a = matrix(rnorm(p.det.long[1] * N[1]), N[1], p.det.long[1]),
                                b = matrix(rnorm(p.det.long[2] * N[2]), N[2], p.det.long[2])),
                       beta = 0)

    # Fit the model.
    out <- intMsPGOcc(occ.formula = ~ occ.cov.1,
                      det.formula = list(f.1 = ~ det.cov.1.1 + det.cov.1.2 + det.cov.1.3,
                                         f.2 = ~ det.cov.2.1 + det.cov.2.2),
                      inits = inits.list,
                      priors = prior.list,
                      data = data.list,
                      n.samples = 100,
                      n.omp.threads = 1,
                      verbose = TRUE,
                      n.report = 10,
                      n.burn = 50,
                      n.thin = 1,
                      n.chains = 1)
    summary(out, level = 'community')
```

---

intPGOcc   *Function for Fitting Single-Species Integrated Occupancy Models Us-*
                        ing Polya-Gamma Latent Variables

Description

Function for fitting single-species integrated occupancy models using Polya-Gamma latent vari-
ables. Data integration is done using a joint likelihood framework, assuming distinct detection
models for each data source that are each conditional on a single latent occurrence process.

Usage

```
intPGOcc(occ.formula, det.formula, data, inits, priors, n.samples,
         n.omp.threads = 1, verbose = TRUE, n.report = 1000,
         n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
         k.fold, k.fold.threads = 1, k.fold.seed,
         k.fold.data, k.fold.only = FALSE, ...)
```

Arguments

occ.formula   a symbolic description of the model to be fit for the occurrence portion of the
                model using R's model syntax. Only right-hand side of formula is specified. See
                example below.

det.formula   a list of symbolic descriptions of the models to be fit for the detection portion
                of the model using R's model syntax for each data set. Each element in the list
                is a formula for the detection model of a given data set. Only right-hand side of
                formula is specified. See example below.

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
                det.covs, and sites. y is a list of matrices or data frames for each data set
                used in the integrated model. Each element of the list has first dimension equal
                to the number of sites with that data source and second dimension equal to the
                maximum number of replicates at a given site. occ.covs is a matrix or data
                frame containing the variables used in the occupancy portion of the model, with
                the number of rows being the number of sites with at least one data source for
                each column (variable). det.covs is a list of variables included in the detection
                portion of the model for each data source. det.covs should have the same num-
                ber of elements as y, where each element is itself a list. Each element of the list
                for a given data source is a different detection covariate, which can be site-level
                or observational-level. Site-level covariates are specified as a vector with length
                equal to the number of observed sites of that data source, while observation-level
                covariates are specified as a matrix or data frame with the number of rows equal
                to the number of observed sites of that data source and number of columns equal
                to the maximum number of replicates at a given site.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta,
                and alpha. The value portion of tags z and beta is the parameter's initial value.
                The tag alpha is a list comprised of the initial values for the detection parame-
                ters for each data source. Each element of the list should be a vector of initial
                values for all detection parameters in the given data source or a single value for
                each data source to assign all parameters for a given data source the same initial
                value. See priors description for definition of each parameter name. Addition-
                ally, 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
                and alpha.normal. Occurrence (beta) and detection (alpha) regression coef-
                ficients are assumed to follow a normal distribution. For beta hyperparameters
                of the normal distribution are passed as a list of length two with the first and sec-
                ond 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. For
                the detection coefficients alpha, the mean and variance hyperparameters are
                themselves passed in as lists, with each element of the list corresponding to the
                specific hyperparameters for the detection parameters in a given data source. If
                not specified, prior means are set to 0 and prior variances set to 2.72.

n.samples   the number of posterior samples to collect in each chain.

n.omp.threads   a positive integer indicating the number of threads to use for SMP parallel pro-
                cessing. 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 sys-
                tems.

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. 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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.data	an integer specifying the specific data set to hold out values from. If not specified, data from all data set locations will be incorporated into the k-fold cross-validation.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the occupancy regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients for all data sources.
z.samples	a coda object of posterior samples for the latent occupancy values
psi.samples	a coda object of posterior samples for the latent occupancy probability values
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().

k.fold.deviance

> scoring rule (deviance) from k-fold cross-validation. A separate deviance value
> is returned for each data source. Only included if k.fold is specified in function
> call. Only a single value is returned if k.fold.data is specified.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribu-
tion is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code
implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.
lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using
Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists.
Ecological monographs, 85(1), 3-28.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian
process models. arXiv preprint arXiv:2001.09111.

Examples

```
set.seed(1008)

# Simulate Data -----------------------------------------------------------
J.x <- 15
J.y <- 15
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 1)
p.occ <- length(beta)
# Detection covariates
```

```
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- runif(2, -1, 1)
}
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
                 n.rep = n.rep, beta = beta, alpha = alpha, sp = FALSE)

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

# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites)

J <- length(dat$z.obs)
# Initial values
inits.list <- list(alpha = list(0, 0, 0, 0),
                   beta = 0,
                   z = rep(1, J))
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = list(0, 0, 0, 0),
                                       var = list(2.72, 2.72, 2.72, 2.72)))
n.samples <- 5000
out <- intPGOcc(occ.formula = ~ occ.cov,
                det.formula = list(f.1 = ~ det.cov.1.1,
                                   f.2 = ~ det.cov.2.1,
                                   f.3 = ~ det.cov.3.1,
                                   f.4 = ~ det.cov.4.1),
                data = data.list,
                inits = inits.list,
                n.samples = n.samples,
                priors = prior.list,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1000,
                n.burn = 1000,
```

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

summary(out)
```

---

lfJSDM   *Function for Fitting a Latent Factor Joint Species Distribution Model*

Description

Function for fitting a joint species distribution model with species correlations. This model does not explicitly account for imperfect detection (see lfMsPGOcc()). We use Polya-gamma latent variables and a factor modeling approach.

Usage

```
lfJSDM(formula, data, inits, priors, n.factors,
       n.samples, n.omp.threads = 1, verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
       k.fold, k.fold.threads = 1, k.fold.seed, k.fold.only = FALSE, ...)
```

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 are allowed using **lme4** syntax (Bates et al. 2015).

data   a list containing data necessary for model fitting. Valid tags are y, covs, and coords. y is a two-dimensional array with first dimension equal to the number of species and second dimension equal to the number of sites. Note how this differs from other spOccupancy functions in that y does not have any replicate surveys. This is because lfJSDM does not account for imperfect detection. covs is a matrix or data frame containing the variables used in the model, with J rows for each column (variable). coords is a matrix with J rows and 2 columns consisting of the spatial coordinates of each site in the data. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

inits   a list with each tag corresponding to a parameter name. Valid tags are beta.comm, beta, tau.sq.beta, sigma.sq.psi, lambda. 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, and sigma.sq.psi.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 set to 2.72. 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 all parameters are assigned the same prior. If not specified, prior shape and scale parameters are set to 0.1. The factor model fits n.factors independent latent factors. The priors for the factor loadings matrix lambda are fixed following standard approaches to ensure parameter identifiability. The upper triangular elements of the N 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.psi is the random effect variance for any random effects, and is 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.

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 0 and N (the number of species in the community). When set to 0, the model assumes there are no residual species correlations, which is equivalent to the msPGOcc() function but without imperfect detection.

n.samples   the number of posterior samples to collect in each chain.

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.

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.

k.fold   specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly

partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.

k.fold.threads   number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.

k.fold.seed   seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.

k.fold.only   a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.

...   currently no additional arguments

Value

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

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

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

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

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

psi.samples   a three-dimensional array of posterior samples for the latent probability of occurrence/detection values for each species.

sigma.sq.psi.samples
                 a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

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

beta.star.samples
                 a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.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().

k.fold.deviance

> vector of scoring rules (deviance) from k-fold cross-validation. A separate value is reported for each species. Only included if `k.fold` is specified in function call.

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()`.

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

Examples

```
set.seed(400)
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- rep(1, J)
N <- 10
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.6, 1.5)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.2, 1.7)
# Detection
# Fix this to be constant and really close to 1.
alpha.mean <- c(9)
tau.sq.alpha <- c(0.05)
p.det <- length(alpha.mean)
# Random effects
# Include a single random effect
psi.RE <- list(levels = c(20),
```

```
                 sigma.sq.psi = c(2))
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
alpha.true <- alpha
# Factor model
factor.model <- TRUE
n.factors <- 4

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                psi.RE = psi.RE, p.RE = p.RE, sp = FALSE,
                factor.model = TRUE, n.factors = 4)

X <- dat$X
y <- dat$y
X.re <- dat$X.re
coords <- dat$coords
occ.covs <- cbind(X, X.re)
colnames(occ.covs) <- c('int', 'occ.cov.1', 'occ.cov.2', 'occ.re.1')
data.list <- list(y = y[, , 1],
                  covs = occ.covs,
                  coords = coords)
# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1))
inits.list <- list(beta.comm = 0, beta = 0, tau.sq.beta = 1)
out <- lfJSDM(formula = ~ occ.cov.1 + occ.cov.2 + (1 | occ.re.1),
              data = data.list,
              inits = inits.list,
              priors = prior.list,
              n.factors = 4,
              n.samples = 1000,
              n.report = 500,
              n.burn = 500,
              n.thin = 2,
              n.chains = 1)
summary(out)
```

---

lfMsPGOcc   *Function for Fitting Latent Factor Multi-Species Occupancy Models*

Description

Function for fitting multi-species occupancy models with species correlations (i.e., a joint species distribution model with imperfect detection). We use Polya-gamma latent variables and a factor modeling approach for dimension reduction.

Usage

```
lfMsPGOcc(occ.formula, det.formula, data, inits, priors, n.factors,
          n.samples, n.omp.threads = 1, verbose = TRUE, n.report = 100,
          n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
          k.fold, k.fold.threads = 1, k.fold.seed, k.fold.only = FALSE, ...)
```

Arguments

occ.formula	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts 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 are allowed using **lme4** syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, and coords. y is a three-dimensional array 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. occ.covs is a matrix or data frame containing the variables used in the occurrence 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 spOccupancy 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, lambda, sigma.sq.psi, sigma.sq.p, z. 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.psi.ig, and sigma.sq.p.ig. Community-level occurrence (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 set to 2.72. Community-level variance parameters for occurrence (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. The factor model fits n.factors independent latent factors. The priors for the factor loadings matrix lambda are fixed following standard approaches to ensure parameter identifiability. The upper triangular elements of the N 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.psi and sigma.sq.p are the random effect variances for any occurrence 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.

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.samples	the number of posterior samples to collect in each chain.
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 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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed

are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k - 1* subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.

k.fold.threads   number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.

k.fold.seed   seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.

k.fold.only   a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.

...   currently no additional arguments

Value

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

beta.comm.samples
                  a coda object of posterior samples for the community level occurrence 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 occurrence 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 occurrence 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.
z.samples   a three-dimensional array of posterior samples for the latent occurrence values for each species.
psi.samples   a three-dimensional array of posterior samples for the latent occurrence probability values for each species.
sigma.sq.psi.samples
                  a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

        a coda object of posterior samples for variances of random intercepts included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

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

beta.star.samples

        a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

        a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in det.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().

k.fold.deviance

        vector of scoring rules (deviance) from k-fold cross-validation. A separate value is reported for each species. Only included if k.fold is specified in function call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

Dorazio, R. M., and Royle, J. A. (2005). Estimating size and composition of biological communities by modeling the occurrence of species. Journal of the American Statistical Association, 100(470), 389-398.

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 <- 8
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- 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, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
p.RE <- list()
# Include a random intercept on detection
p.RE <- list(levels = c(40),
             sigma.sq.p = c(2))
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 4

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                sp = FALSE, factor.model = TRUE, n.factors = n.factors, p.RE = p.RE)
y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.p.re <- dat$X.p.re
# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3],
                 det.re = X.p.re[, , 1])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = dat$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
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
                   beta = 0,
                   alpha = 0,
                   tau.sq.beta = 1,
                   tau.sq.alpha = 1,
                   lambda = lambda.inits,
                   z = apply(y, c(1, 2), max, na.rm = TRUE))

n.samples <- 300
n.burn <- 200
n.thin <- 1

out <- lfMsPGOcc(occ.formula = ~ occ.cov,
                 det.formula = ~ det.cov.1 + det.cov.2 + (1 | det.re),
                 data = data.list,
                 inits = inits.list,
                 n.samples = n.samples,
                 priors = prior.list,
                 n.factors = n.factors,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 n.report = 100,
                 n.burn = n.burn,
                 n.thin = n.thin,
                 n.chains = 1)

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

---

msPGOcc   *Function for Fitting Multi-Species Occupancy Models Using Polya-*
                                Gamma Latent Variables

Description

Function for fitting multi-species occupancy models using Polya-Gamma latent variables.

Usage

```
msPGOcc(occ.formula, det.formula, data, inits, priors, n.samples,
        n.omp.threads = 1, verbose = TRUE, n.report = 100,
```

```
                n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
                k.fold, k.fold.threads = 1, k.fold.seed, k.fold.only = FALSE, ...)
```

Arguments

`occ.formula`	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts 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 are allowed using **lme4** syntax (Bates et al. 2015).
`data`	a list containing data necessary for model fitting. Valid tags are `y`, `occ.covs`, and `det.covs`. `y` is a three-dimensional array 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. `occ.covs` is a matrix or data frame containing the variables used in the occurrence 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.
`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.psi`, `sigma.sq.p`, and `z`. 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.psi.ig`, and `sigma.sq.p.ig`. Community-level occurrence (`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 set to 2.72. Community-level variance parameters for occurrence (`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.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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.

n.samples	the number of posterior samples to collect in each chain.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and `k.fold.threads` and `k.fold.seed` are ignored. In *k*-fold cross-validation, the data specified in `data` is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the `k.fold.deviance` object in the return list.
k.fold.threads	number of threads to use for cross-validation. If `k.fold.threads > 1` parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if `k.fold` is not specified.
k.fold.seed	seed used to split data set into `k.fold` parts for k-fold cross-validation. Ignored if `k.fold` is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (`TRUE`) or perform cross-validation after fitting the full model (`FALSE`). Default value is `FALSE`.
...	currently no additional arguments

Value

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

beta.comm.samples

a `coda` object of posterior samples for the community level occurrence 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 occurrence 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 occurrence regression coefficients.

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

z.samples   a three-dimensional array of posterior samples for the latent occurrence values for each species.

psi.samples   a three-dimensional array of posterior samples for the latent occurrence probability values for each species.

sigma.sq.psi.samples

>   a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

>   a coda object of posterior samples for variances of random intercepts included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

>   a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

>   a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in det.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().

k.fold.deviance

>   vector of scoring rules (deviance) from k-fold cross-validation. A separate value is reported for each species. Only included if k.fold is specified in function call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

Dorazio, R. M., and Royle, J. A. (2005). Estimating size and composition of biological communities by modeling the occurrence of species. Journal of the American Statistical Association, 100(470), 389-398.

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 <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- 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, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
```

```
dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                sp = FALSE)
y <- dat$y
X <- dat$X
X.p <- dat$X.p
# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  occ.covs = occ.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,
                   z = apply(y, c(1, 2), max, na.rm = TRUE))

n.samples <- 3000
n.burn <- 2000
n.thin <- 1

out <- msPGOcc(occ.formula = ~ occ.cov,
               det.formula = ~ det.cov.1 + det.cov.2,
               data = data.list,
               inits = inits.list,
               n.samples = n.samples,
               priors = prior.list,
               n.omp.threads = 1,
               verbose = TRUE,
               n.report = 1000,
               n.burn = n.burn,
               n.thin = n.thin,
               n.chains = 1)

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

neon2015	*Detection-nondetection data of 12 foliage gleaning bird species in 2015 in Bartlett Experimental Forest in New Hampshire, USA*

Description

Detection-nondetection data of 12 foliage gleaning bird species in 2015 in the Bartlett Experimental Forest in New Hampshire, USA. These data were collected as part of the National Ecological Observatory Network (NEON). Data were collected at 80 sites where observers recorded the number of all bird species observed during a six minute, 125m radius point count survey once during the breeding season. The six minute survey was split into three two-minute intervals following a removal design where the observer recorded the interval during which a species was first observed (if any) with a 1, intervals prior to observation with a 0, and then mentally removed the species from subsequent intervals (marked with NA), which enables modeling of data in an occupancy modeling framework. 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(neon2015)
```

Format

neon2015 is a list with four elements:

y: a three-dimensional array of detection-nondetection data with dimensions of species (12), sites (80) and replicates (3).

occ.covs: a numeric matrix with 80 rows and one column consisting of the elevation at each site.

det.covs: a list of two numeric vectors with 80 elements. The first element is the day of year when the survey was conducted for a given site. The second element is the time of day when the survey began.

coords: a numeric matrix with 80 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

NEON (National Ecological Observatory Network). Breeding landbird point counts, RELEASE-2021 (DP1.10003.001). https://doi.org/10.48443/s730-dy13. Dataset accessed from https://data.neonscience.org on October 10, 2021

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

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.

PGOcc	*Function for Fitting Single-Species Occupancy Models Using Polya-Gamma Latent Variables*

Description

Function for fitting single-species occupancy models using Polya-Gamma latent variables.

Usage

```
PGOcc(occ.formula, det.formula, data, inits, priors, n.samples,
      n.omp.threads = 1, verbose = TRUE, n.report = 100,
      n.burn = round(.10 * n.samples), n.thin = 1, n.chains = 1,
      k.fold, k.fold.threads = 1, k.fold.seed, k.fold.only = FALSE, ...)
```

Arguments

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

data
: a list containing data necessary for model fitting. Valid tags are y, occ.covs, and det.covs. y is 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. occ.covs is a matrix or data frame containing the variables used in the occurrence 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.

inits
: a list with each tag corresponding to a parameter name. Valid tags are z, beta, alpha, sigma.sq.psi, and sigma.sq.p. The value portion of each tag is the parameter's initial value. sigma.sq.psi and sigma.sq.p are only relevant when including random effects in the occurrence and detection portion of the occupancy model, respectively. 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, sigma.sq.psi.ig, and sigma.sq.p.ig. Occupancy (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 2.72. `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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.
n.samples	the number of posterior samples to collect in each chain.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples   a coda object of posterior samples for the occupancy regression coefficients.

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

z.samples   a coda object of posterior samples for the latent occupancy values

psi.samples   a coda object of posterior samples for the latent occupancy probability values

sigma.sq.psi.samples

    a coda object of posterior samples for variances of random intercepts included in the occupancy portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

    a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

    a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

    a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in det.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().

k.fold.deviance

    scoring rule (deviance) from k-fold cross-validation. Only included if k.fold is specified in function call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm. 2002. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology 83: 2248-2255.

Examples

```
set.seed(400)
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, -0.15)
p.occ <- length(beta)
alpha <- c(0.7, 0.4)
p.det <- length(alpha)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sp = FALSE)
occ.covs <- dat$X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov = dat$X.p[, , 2])
# Data bundle
data.list <- list(y = dat$y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
                   z = apply(data.list$y, 1, max, na.rm = TRUE))

n.samples <- 5000
n.report <- 1000

out <- PGOcc(occ.formula = ~ occ.cov,
             det.formula = ~ det.cov,
             data = data.list,
             inits = inits.list,
             n.samples = n.samples,
             priors = prior.list,
             n.omp.threads = 1,
             verbose = TRUE,
```

```
                    n.report = n.report,
                    n.burn = 1000,
                    n.thin = 1,
                    n.chains = 1)
  summary(out)
```

---

postHocLM   *Function for Fitting Linear Mixed Models Previous Model Estimates*

Description

Function for fitting a linear (mixed) model as a second-stage model where the response variable itself comes from a previous model fit and has uncertainty associated with it. The response variable is assumed to be a set of estimates from a previous model fit, where each value in the response variable has a posterior MCMC sample of estimates. This function is useful for doing "posthoc" analyses of model estimates (e.g., exploring how species traits relate to species-specific parameter estimates from a multi-species occupancy model). Such analyses are sometimes referred to as "two-stage" analyses.

Usage

```
postHocLM(formula, data, inits, priors, verbose = FALSE,
  n.report = 100, n.samples, 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 are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y and covs. y is a matrix or data frame with first dimension equal to the number of posterior samples of each value in the response variable and the second dimension is equal to the number of values in the response variable. For example, if the response is species-specific covariate effect estimates from a multi-species occupancy model, the rows correspond to the posterior MCMC samples and the columns correspond to species. covs is a matrix or data frame containing the independent variables used in the model. Note the number of rows of covs should be equal to the number of columns in y.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta, tau.sq, and sigma.sq. The value portion of each tag is the parameter's initial value. sigma.sq is only relevant when including random effects in the model. 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`,
                `tau.sq.ig`, and `sigma.sq.ig`. Regression coefficients (`beta`) 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. `tau.sq` is the residual variance, and is
                assumed to follow an inverse-Gamma distribution. The hyperparameters of the
                inverse-Gamma distribution are passed as a vector of length two with first and
                second elements corresponding to the shape and scale parameters, respectively.
                `sigma.sq` are the variances of any random intercepts included in the model,
                which similarly to `tau.sq` follow an inverse-Gamma distribution. The hyper-
                parameters 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.

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.samples   the number of posterior samples to collect in each chain. Note that by default,
                the same number of MCMC samples fit in the first stage model is assumed to be
                fit for the second stage model. If `n.samples` is specified, it must be a multiple of
                the number of samples fit in the first stage, otherwise an error will be reported.

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

...   currently no additional arguments

Value

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

beta.samples   a coda object of posterior samples for the regression coefficients.

tau.sq.samples a coda object of posterior samples for the residual variances.

y.hat.samples   a coda object of posterior samples of fitted values.

sigma.sq.samples
                a coda object of posterior samples for the random effect variances if any random
                intercepts were included in the model.

beta.star.samples
                a coda object of posterior samples for the random effects. Only included if
                random intercepts are specified in `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()`.

bayes.R2   a coda object of posterior samples of the Bayesian R-squared as a measure of
                  model fit. Note that when random intercepts are included in the model, this is
                  the conditional Bayesian R-squared, not the marginal Bayesian R-squared.

The return object will include additional objects used for subsequent summarization.

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.

Examples

```
# Simulate Data ------------------------------------------------------------
set.seed(100)
N <- 100
beta <- c(0, 0.5, 1.2)
tau.sq <- 1
p <- length(beta)
X <- matrix(1, nrow = N, ncol = p)
if (p > 1) {
  for (i in 2:p) {
    X[, i] <- rnorm(N)
  } # i
}
mu <- X
y <- rnorm(N, mu, sqrt(tau.sq))
# Replicate y n.samples times and add a small amount of noise that corresponds
# to uncertainty from a first stage model.
n.samples <- 1000
y <- matrix(y, n.samples, N, byrow = TRUE)
y <- y + rnorm(length(y), 0, 0.25)

# Package data for use with postHocLM -------------------------------------
colnames(X) <- c('int', 'cov.1', 'cov.2')
data.list <- list(y = y,
  covs = X)
data <- data.list
inits <- list(beta = 0, tau.sq = 1)
priors <- list(beta.normal = list(mean = 0, var = 10000),
       tau.sq.ig = c(0.001, 0.001))

# Run the model -----------------------------------------------------------
out <- postHocLM(formula = ~ cov.1 + cov.2,
 inits = inits,
 data = data.list,
 priors = priors,
```

```
  verbose = FALSE,
  n.chains = 1)
summary(out)
```

---

ppcOcc   *Function for performing posterior predictive checks*

Description

Function for performing posterior predictive checks on spOccupancy model objects.

Usage

```
ppcOcc(object, fit.stat, group, ...)
```

Arguments

object	an object of class PGOcc, spPGOcc, msPGOcc, spMsPGOcc, intPGOcc, spIntPGOcc, lfMsPGOcc, sfMsPGOcc, tPGOcc, stPGOcc, svcPGOcc, svcMsPGOcc, tMsPGOcc, stMsPGOcc, svcTMsPGOcc.
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 detection-nondetection data for the posterior predictive check. Value 1 will group values by row (site) and value 2 will group values by column (replicate).
...	currently no additional arguments

Details

Standard GoF assessments are not valid for binary data, and posterior predictive checks must be performed on some sort of binned data.

Value

An object of class ppcOcc 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 PGOcc, spPGOcc, or svcPGOcc. When object is of class msPGOcc, spMsPGOcc, lfMsPGOcc, sfMsPGOcc, or svcMsPGOcc this is a numeric matrix with rows corresponding to posterior samples and columns corresponding to species. When object is of class intPGOcc or spIntPGOcc, this is a list, with each element of the list being a vector of posterior samples for each data set. When object is of class tPGOcc or stPGOcc, this is a numeric matrix with rows corresponding to posterior samples and columns corresponding to primary sampling periods. When object is of class tMsPGOcc, stMsPGOcc, or svcTMsPGOcc, this is a three-dimensional array with dimensions corresponding to MCMC sample, species, and primary time period.

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 `PGOcc`, `spPGOcc`, or
                   `svcPGOcc`. When `object` is of class `msPGOcc`, `spMsPGOcc`, `lfMsPGOcc`, `sfMsPGOcc`,
                   or `svcMsPGOcc` this is a numeric matrix with rows corresponding to posterior
                   samples and columns corresponding to species. When `object` is of class `intPGOcc`
                   or `spIntPGOcc`, this is a list, with each element of the list being a vector of poste-
                   rior samples for each data set. When `object` is of class `tPGOcc` or `stPGOcc`, this
                   is a numeric matrix with rows corresponding to posterior samples and columns
                   corresponding to primary sampling periods. When `object` is of class `tMsPGOcc`,
                   `stMsPGOcc`, or `svcTMsPGOcc`, this is a three-dimensional array with dimensions
                   corresponding to MCMC sample, species, and primary time period.

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., sites when
                   group = 1, replicates when group = 2) when `object` is of class `PGOcc`, `spPGOcc`,
                   or `svcPGOcc`. When `object` is of class `msPGOcc`, `spMsPGOcc`, `lfMsPGOcc`, `sfMsPGOcc`,
                   `svcMsPGOcc`, this is a three-dimensional array with the additional dimension cor-
                   responding to species. When `object` is of class `intPGOcc` or `spIntPGOcc`, this is
                   a list, with each element consisting of the posterior quantile matrix for each data
                   set. When `object` is of class `tPGOcc` or `stPGOcc`, this is a three-dimensional
                   array with the additional dimension corresponding to primary sampling periods.
                   When `object` is of class `tMsPGOcc`, `stMsPGOcc`, `svcTMsPGOcc`, this is a four-
                   dimensional array with dimensions corresponding to quantile, species, grouping
                   element, and primary time period.

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.,
                   sites when group = 1, replicates when group = 2) when `object` is of class
                   `PGOcc`, `spPGOcc`, `svcPGOcc`. When `object` is of class `msPGOcc`, `spMsPGOcc`,
                   `lfMsPGOcc`, `sfMsPGOcc`, or `svcMsPGOcc`, this is a three-dimensional array with
                   the additional dimension corresponding to species. When `object` is of class
                   `intPGOcc` or `spIntPGOcc`, this is a list, with each element consisting of the pos-
                   terior quantile matrix for each data set. When `object` is of class `tPGOcc` or
                   `stPGOcc`, this is a three-dimensional array with the additional dimension cor-
                   responding to primary sampling periods. When `object` is of class `tMsPGOcc`,
                   `stMsPGOcc`, `svcTMsPGOcc`, this is a four-dimensional array with dimensions cor-
                   responding to quantile, species, grouping element, and primary time period.

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

Author(s)

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

Examples

```
set.seed(400)
# Simulate Data -----------------------------------------------------------
```

```
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, -0.15)
p.occ <- length(beta)
alpha <- c(0.7, 0.4)
p.det <- length(alpha)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sp = FALSE)
occ.covs <- dat$X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov = dat$X.p[, , 2])
# Data bundle
data.list <- list(y = dat$y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)

# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
                   z = apply(data.list$y, 1, max, na.rm = TRUE))

n.samples <- 5000
n.report <- 1000

out <- PGOcc(occ.formula = ~ occ.cov,
             det.formula = ~ det.cov,
             data = data.list,
             inits = inits.list,
             n.samples = n.samples,
             priors = prior.list,
             n.omp.threads = 1,
             verbose = TRUE,
             n.report = n.report,
             n.burn = 4000,
             n.thin = 1)

# Posterior predictive check
ppc.out <- ppcOcc(out, fit.stat = 'chi-squared', group = 1)
summary(ppc.out)
```

---

predict.intMsPGOcc   *Function for prediction at new locations for integrated multi-species occupancy models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'intMsPGOcc'. Prediction is currently possible only for the latent occupancy state.

Usage

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

Arguments

`object`	an object of class intMsPGOcc
`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 occupancy (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 `intMsPGOcc`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `intMsPGOcc`. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of `intMsPGOcc`.
`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

Value

A list object of class `predict.intMsPGOcc` consisting of:

`psi.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
`z.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence 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(91)
J.x <- 10
J.y <- 10
# Total number of data sources across the study region
J.all <- J.x * J.y
# Number of data sources.
n.data <- 2
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
n.rep <- list()
n.rep[[1]] <- rep(3, J.obs[1])
n.rep[[2]] <- rep(4, J.obs[2])

# Number of species observed in each data source
N <- c(8, 3)

# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.4, 0.3)
# Detection
# Detection covariates
alpha.mean <- list()
tau.sq.alpha <- list()
# Number of detection parameters in each data source
p.det.long <- c(4, 3)
for (i in 1:n.data) {
  alpha.mean[[i]] <- runif(p.det.long[i], -1, 1)
  tau.sq.alpha[[i]] <- runif(p.det.long[i], 0.1, 1)
}
# Random effects
psi.RE <- list()
p.RE <- list()
beta <- matrix(NA, nrow = max(N), ncol = p.occ)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(max(N), beta.mean[i], sqrt(tau.sq.beta[i]))
}
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- matrix(NA, nrow = N[i], ncol = p.det.long[i])
  for (t in 1:p.det.long[i]) {
    alpha[[i]][, t] <- rnorm(N[i], alpha.mean[[i]][t], sqrt(tau.sq.alpha[[i]])[t])
  }
}
sp <- FALSE
```

```
factor.model <- FALSE
# Simulate occupancy data
dat <- simIntMsOcc(n.data = n.data, J.x = J.x, J.y = J.y,
   J.obs = J.obs, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
            psi.RE = psi.RE, p.RE = p.RE, sp = sp, factor.model = factor.model,
                     n.factors = n.factors)
J <- nrow(dat$coords.obs)
y <- dat$y
X <- dat$X.obs
X.p <- dat$X.p
X.re <- dat$X.re.obs
X.p.re <- dat$X.p.re
sites <- dat$sites
species <- dat$species

# Package all data into a list
occ.covs <- cbind(X)
colnames(occ.covs) <- c('int', 'occ.cov.1')
#colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2],
      det.cov.1.2 = X.p[[1]][, , 3],
      det.cov.1.3 = X.p[[1]][, , 4])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2],
      det.cov.2.2 = X.p[[2]][, , 3])

data.list <- list(y = y,
  occ.covs = occ.covs,
  det.covs = det.covs,
                   sites = sites,
                   species = species)
# Take a look at the data.list structure for integrated multi-species
# occupancy models.
# Priors
prior.list <- list(beta.comm.normal = list(mean = 0,
      var = 2.73),

  alpha.comm.normal = list(mean = list(0, 0),
                    var = list(2.72, 2.72)),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   tau.sq.alpha.ig = list(a = list(0.1, 0.1),
 b = list(0.1, 0.1)))
inits.list <- list(alpha.comm = list(0, 0),
  beta.comm = 0,
  tau.sq.beta = 1,
  tau.sq.alpha = list(1, 1),
             alpha = list(a = matrix(rnorm(p.det.long[1] * N[1]), N[1], p.det.long[1]),
 b = matrix(rnorm(p.det.long[2] * N[2]), N[2], p.det.long[2])),
  beta = 0)

# Fit the model.
out <- intMsPGOcc(occ.formula = ~ occ.cov.1,
```

```
                det.formula = list(f.1 = ~ det.cov.1.1 + det.cov.1.2 + det.cov.1.3,
                                   f.2 = ~ det.cov.2.1 + det.cov.2.2),
 inits = inits.list,
 priors = prior.list,
         data = data.list,
         n.samples = 100,
                 n.omp.threads = 1,
         verbose = TRUE,
         n.report = 10,
         n.burn = 50,
         n.thin = 1,
         n.chains = 1)
#Predict at new locations.
X.0 <- dat$X.pred
psi.0 <- dat$psi.pred
out.pred <- predict(out, X.0, ignore.RE = TRUE)

# Create prediction for one species.
curr.sp <- 2
psi.hat.quants <- apply(out.pred$psi.0.samples[,curr.sp, ],
2, quantile, c(0.025, 0.5, 0.975))
plot(psi.0[curr.sp, ], psi.hat.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Predicted', ylim = c(min(psi.hat.quants), max(psi.hat.quants)),
     main = paste("Species ", curr.sp, sep = ''))
segments(psi.0[curr.sp, ], psi.hat.quants[1, ], psi.0[curr.sp, ], psi.hat.quants[3, ])
lines(psi.0[curr.sp, ], psi.0[curr.sp, ])
```

---

predict.intPGOcc	*Function for prediction at new locations for single-species integrated occupancy models*

Description

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

Usage

```
## S3 method for class 'intPGOcc'
predict(object, X.0, ...)
```

Arguments

object	an object of class intPGOcc
X.0	the design matrix for prediction locations. This should include a column of 1s for the intercept. Covariates should have the same column names as those used when fitting the model with intPGOcc.
...	currently no additional arguments

Value

An object of class `predict.intPGOcc` that is a list comprised of:

`psi.0.samples` a coda object of posterior predictive samples for the latent occurrence probability values.

`z.0.samples` a coda object of posterior predictive samples for the latent occurrence values.

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

Author(s)

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

Examples

```
set.seed(1008)

# Simulate Data -------------------------------------------------------------
J.x <- 10
J.y <- 10
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 1)
p.occ <- length(beta)
# Detection covariates
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- runif(2, -1, 1)
}
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
                 n.rep = n.rep, beta = beta, alpha = alpha, sp = FALSE)

y <- dat$y
X <- dat$X.obs
X.p <- dat$X.p
sites <- dat$sites
```

```
# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites)

J <- length(dat$z.obs)
# Initial values
inits.list <- list(alpha = list(0, 0, 0, 0),
                   beta = 0,
                   z = rep(1, J))
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = list(0, 0, 0, 0),
                                       var = list(2.72, 2.72, 2.72, 2.72)))
n.samples <- 5000
out <- intPGOcc(occ.formula = ~ occ.cov,
                det.formula = list(f.1 = ~ det.cov.1.1,
                                   f.2 = ~ det.cov.2.1,
                                   f.3 = ~ det.cov.3.1,
                                   f.4 = ~ det.cov.4.1),
                data = data.list,
                inits = inits.list,
                n.samples = n.samples,
                priors = prior.list,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1000,
                n.burn = 4000,
                n.thin = 1)

summary(out)

# Prediction
X.0 <- dat$X.pred
psi.0 <- dat$psi.pred

out.pred <- predict(out, X.0)
psi.hat.quants <- apply(out.pred$psi.0.samples, 2, quantile, c(0.025, 0.5, 0.975))
plot(psi.0, psi.hat.quants[2, ], pch = 19, xlab = 'True',
     ylab = 'Fitted', ylim = c(min(psi.hat.quants), max(psi.hat.quants)))
segments(psi.0, psi.hat.quants[1, ], psi.0, psi.hat.quants[3, ])
lines(psi.0, psi.0)
```

---

predict.lfJSDM   *Function for prediction at new locations for latent factor joint species distribution models*

Description

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

Usage

```
## S3 method for class 'lfJSDM'
predict(object, X.0, coords.0,
        ignore.RE = FALSE, ...)
```

Arguments

`object`	an object of class lfJSDM
`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 `lfJSDM`. Columns should correspond to the order of how covariates were specified in the formula argument of `lfJSDM`. Column names of the random effects must match the name of the random effects, if specified in the formula argument of `lfJSDM`.
`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.
`...`	currently no additional arguments

Value

A list object of class `predict.lfJSDM` that consists of:

`psi.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
`z.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence values.
`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 sup-
ported for prediction. For sampled levels, the posterior distribution for the random intercept corre-
sponding 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 <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- 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, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}

n.factors <- 3
dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                sp = FALSE, factor.model = TRUE, n.factors = n.factors)
n.samples <- 5000
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
# Summarize the multiple replicates into a single value for use in a JSDM
y <- apply(dat$y[, -pred.indx, ], c(1, 2), max, na.rm = TRUE)
# Covariates
X <- dat$X[-pred.indx, ]
# Spatial coordinates
```

```
coords <- dat$coords[-pred.indx, ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
psi.0 <- dat$psi[, pred.indx]
coords.0 <- dat$coords[pred.indx, ]
# Package all data into a list
covs <- X[, 2, drop = FALSE]
colnames(covs) <- c('occ.cov')
data.list <- list(y = y,
                  covs = covs,
                  coords = coords)

# Occupancy initial values
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1))
# Initial values
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
                   beta = 0,
                   tau.sq.beta = 1,
                   lambda = lambda.inits)

out <- lfJSDM(formula = ~ occ.cov,
              data = data.list,
              inits = inits.list,
              n.samples = n.samples,
              n.factors = 3,
              priors = prior.list,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = 1000,
              n.burn = 4000)

summary(out)

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

---

predict.lfMsPGOcc	*Function for prediction at new locations for latent factor multi-species occupancy models*

Description

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

Usage

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

Arguments

`object`	an object of class lfMsPGOcc
`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 occupancy (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 `lfMsPGOcc`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `lfMsPGOcc`. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of `lfMsPGOcc`.
`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.
`...`	currently no additional arguments
`type`	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

Value

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

`psi.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
`z.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence values.
`w.0.samples`	a three-dimensional array of posterior predictive samples for the 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 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(400)
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep<- sample(2:4, size = J, replace = TRUE)
N <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- 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, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}

n.factors <- 3
dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                sp = FALSE, factor.model = TRUE, n.factors = n.factors)
n.samples <- 5000
# 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, ]
# Spatial coordinates
coords <- dat$coords[-pred.indx, ]
```

```
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
psi.0 <- dat$psi[, pred.indx]
coords.0 <- dat$coords[pred.indx, ]
# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  occ.covs = occ.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
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
                   beta = 0,
                   alpha = 0,
                   tau.sq.beta = 1,
                   tau.sq.alpha = 1,
                   lambda = lambda.inits,
                   z = apply(y, c(1, 2), max, na.rm = TRUE))

out <- lfMsPGOcc(occ.formula = ~ occ.cov,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.samples = n.samples,
                 n.factors = 3,
                 priors = prior.list,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 n.report = 1000,
                 n.burn = 4000)

summary(out, level = 'community')

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

predict.msPGOcc *Function for prediction at new locations for multi-species occupancy models*

Description

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

Usage

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

Arguments

`object`	an object of class msPGOcc
`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 occupancy (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 `msPGOcc`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `msPGOcc`. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of `msPGOcc`.
`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 'occupancy' to predict latent occupancy probability and latent occupancy values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.

Value

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

`psi.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
`z.0.samples`	a three-dimensional array of posterior predictive samples for the latent occurrence values.

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

p.0.samples   a three-dimensional array of posterior predictive samples for the detection prob-
                   ability 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 sup-
ported for prediction. For sampled levels, the posterior distribution for the random intercept corre-
sponding 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 <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- 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, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                sp = FALSE)
n.samples <- 5000
# 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, ]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
psi.0 <- dat$psi[, pred.indx]
# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  occ.covs = occ.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,
                   z = apply(y, c(1, 2), max, na.rm = TRUE))

out <- msPGOcc(occ.formula = ~ occ.cov,
               det.formula = ~ det.cov.1 + det.cov.2,
               data = data.list,
               inits = inits.list,
               n.samples = n.samples,
               priors = prior.list,
               n.omp.threads = 1,
               verbose = TRUE,
               n.report = 1000,
               n.burn = 4000)

summary(out, level = 'community')

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

---

predict.PGOcc   *Function for prediction at new locations for single-species occupancy*
                                 models

Description

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

Usage

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

Arguments

`object`	an object of class PGOcc
`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 occupancy (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 PGOcc. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of PGOcc. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of PGOcc.
`ignore.RE`	logical value that specifies whether or not to remove random occurrence (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 'occupancy' to predict latent occupancy probability and latent occupancy 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.PGOcc`. When `type = 'occupancy'`, the list consists of:

`psi.0.samples`	a coda object of posterior predictive samples for the latent occupancy probability values.
`z.0.samples`	a coda object of posterior predictive samples for the latent occupancy values.

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(400)
# 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.occ <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
dat <- simOcc(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$y[-pred.indx, ]
# Occupancy covariates
X <- dat$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
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov = X.p[, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)
# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.occ),
                                      var = rep(2.72, p.occ)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)))
# Initial values
inits.list <- list(alpha = rep(0, p.det),
```

```
                        beta = rep(0, p.occ),
                        z = apply(y, 1, max, na.rm = TRUE))

n.samples <- 5000
n.report <- 1000

out <- PGOcc(occ.formula = ~ occ.cov,
             det.formula = ~ det.cov,
             data = data.list,
             inits = inits.list,
             n.samples = n.samples,
             priors = prior.list,
             n.omp.threads = 1,
             verbose = TRUE,
             n.report = n.report,
             n.burn = 4000,
             n.thin = 1)

summary(out)

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

---

predict.sfJSDM   *Function for prediction at new locations for spatial factor joint species*
                         distribution model

Description

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

Usage

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

Arguments

object   an object of class sfJSDM

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 sfJSDM. Columns should correspond to the order of how covariates were specified in the formula argument of sfJSDM. Column names of the random effects must match the name of the random effects, if specified in the formula argument of sfJSDM.
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, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.
...	currently no additional arguments

Value

An list object of class predict.sfJSDM that consists of:

psi.0.samples	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
z.0.samples	a three-dimensional array of posterior predictive samples for the latent occurrence values.
w.0.samples	a three-dimensional array of posterior predictive samples for the latent spatial factors.
run.time	execution time reported using proc.time().

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(400)

# Simulate Data -----------------------------------------------------------
J.x <- 7
J.y <- 7
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
N <- 5
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -.2)
tau.sq.alpha <- c(0.2, 0.3, 0.8)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 3
phi <- runif(n.factors, 3/1, 3/.4)
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                phi = phi, sigma.sq = sigma.sq, sp = TRUE, cov.model = 'exponential',
                factor.model = TRUE, n.factors = n.factors)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.samples <- n.batch * batch.length

# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
# Summarize the multiple replicates into a single value for use in a JSDM
y <- apply(dat$y[, -pred.indx, ], c(1, 2), max, na.rm = TRUE)
# Occupancy covariates
X <- dat$X[-pred.indx, ]
```

```
# Coordinates
coords <- as.matrix(dat$coords[-pred.indx, ])
# Prediction values
X.0 <- dat$X[pred.indx, ]
coords.0 <- as.matrix(dat$coords[pred.indx, ])
psi.0 <- dat$psi[, pred.indx]

# Package all data into a list
covs <- X[, 2, drop = FALSE]
colnames(covs) <- c('occ.cov')
data.list <- list(y = y,
                  covs = covs,
                  coords = coords)

# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   phi.unif = list(a = 3/1, b = 3/.1))
# Starting values
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   tau.sq.beta = 1,
                   phi = 3 / .5,
                   sigma.sq = 2,
                   lambda = lambda.inits)
# Tuning
tuning.list <- list(phi = 1)

out <- sfJSDM(formula = ~ occ.cov,
              data = data.list,
              inits = inits.list,
              n.batch = n.batch,
              batch.length = batch.length,
              accept.rate = 0.43,
              n.factors = 3,
              priors = prior.list,
              cov.model = "exponential",
              tuning = tuning.list,
              n.omp.threads = 1,
              verbose = TRUE,
              NNGP = TRUE,
              n.neighbors = 5,
              search.type = 'cb',
              n.report = 10,
              n.burn = 100,
              n.thin = 1)

summary(out, level = 'both')

# Predict at new locations -------------------------------------------
```

```
out.pred <- predict(out, X.0, coords.0, verbose = FALSE)
```

---

predict.sfMsPGOcc   *Function for prediction at new locations for spatial factor multi-*
species occupancy models

Description

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

Usage

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

Arguments

`object`	an object of class sfMsPGOcc
`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 occupancy (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 `sfMsPGOcc`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `sfMsPGOcc`. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of `sfMsPGOcc`.
`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, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.

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

...   currently no additional arguments

Value

An list object of class `predict.sfMsPGOcc`. When `type = 'occupancy'`, the list consists of:

psi.0.samples a three-dimensional array of posterior predictive samples for the latent occur-
               rence probability values.

z.0.samples   a three-dimensional array of posterior predictive samples for the latent occur-
               rence values.

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

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

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

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

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

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 sup-
ported for prediction. For sampled levels, the posterior distribution for the random intercept corre-
sponding 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)

# Simulate Data -------------------------------------------------------
J.x <- 7
J.y <- 7
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
```

```
N <- 5
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -.2)
tau.sq.alpha <- c(0.2, 0.3, 0.8)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 3
phi <- runif(n.factors, 3/1, 3/.4)
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                phi = phi, sigma.sq = sigma.sq, sp = TRUE, cov.model = 'exponential',
                factor.model = TRUE, n.factors = n.factors)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.samples <- n.batch * batch.length

# 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, ]
# Coordinates
coords <- as.matrix(dat$coords[-pred.indx, ])
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
coords.0 <- as.matrix(dat$coords[pred.indx, ])
psi.0 <- dat$psi[, pred.indx]

# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
```

```
                        occ.covs = occ.covs,
                        det.covs = det.covs,
                        coords = coords)

# Priors
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, b = 3/.1),
                    sigma.sq.ig = list(a = 2, b = 2))
# Starting values
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))
inits.list <- list(alpha.comm = 0,
                    beta.comm = 0,
                    beta = 0,
                    alpha = 0,
                    tau.sq.beta = 1,
                    tau.sq.alpha = 1,
                    phi = 3 / .5,
                    sigma.sq = 2,
                    lambda = lambda.inits,
                    z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- sfMsPGOcc(occ.formula = ~ occ.cov,
                  det.formula = ~ det.cov.1 + det.cov.2,
                  data = data.list,
                  inits = inits.list,
                  n.batch = n.batch,
                  batch.length = batch.length,
                  accept.rate = 0.43,
                  n.factors = 3,
                  priors = prior.list,
                  cov.model = "exponential",
                  tuning = tuning.list,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  NNGP = TRUE,
                  n.neighbors = 5,
                  search.type = 'cb',
                  n.report = 10,
                  n.burn = 100,
                  n.thin = 1)

summary(out, level = 'both')

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

predict.spIntPGOcc	*Function for prediction at new locations for single-species integrated spatial occupancy models*

Description

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

Usage

```
## S3 method for class 'spIntPGOcc'
predict(object, X.0, coords.0, n.omp.threads = 1, verbose = TRUE,
        n.report = 100, ...)
```

Arguments

object	an object of class `spIntPGOcc`.
X.0	the design matrix for prediction locations. This should include a column of 1s for the intercept. Covariates should have the same column names as those used when fitting the model with `spIntPGOcc`.
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.
...	currently no additional arguments

Value

An object of class `predict.spIntPGOcc` that is a list comprised of:

psi.0.samples	a coda object of posterior predictive samples for the latent occurrence probability values.
z.0.samples	a coda object of posterior predictive samples for the latent occurrence values.

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

Author(s)

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

References

Hooten, M. B., and Hefley, T. J. (2019). Bringing Bayesian models to life. CRC Press.

Examples

```
set.seed(400)

# Simulate Data -------------------------------------------------------
# Number of locations in each direction. This is the total region of interest
# where some sites may or may not have a data source.
J.x <- 8
J.y <- 8
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 0.5)
p.occ <- length(beta)
# Detection covariates
alpha <- list()
alpha[[1]] <- runif(2, 0, 1)
alpha[[2]] <- runif(3, 0, 1)
alpha[[3]] <- runif(2, -1, 1)
alpha[[4]] <- runif(4, -1, 1)
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)
sigma.sq <- 2
phi <- 3 / .5
sp <- TRUE

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
                 n.rep = n.rep, beta = beta, alpha = alpha, sp = sp,
                 phi = phi, sigma.sq = sigma.sq, cov.model = 'spherical')

y <- dat$y
X <- dat$X.obs
X.p <- dat$X.p
sites <- dat$sites
X.0 <- dat$X.pred
psi.0 <- dat$psi.pred
coords <- as.matrix(dat$coords.obs)
coords.0 <- as.matrix(dat$coords.pred)

# Package all data into a list
```

```
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2],
                      det.cov.2.2 = X.p[[2]][, , 3])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2],
                      det.cov.4.2 = X.p[[4]][, , 3],
                      det.cov.4.3 = X.p[[4]][, , 4])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites,
                  coords = coords)

J <- length(dat$z.obs)

# Initial values
inits.list <- list(alpha = list(0, 0, 0, 0),
                   beta = 0,
                   phi = 3 / .5,
                   sigma.sq = 2,
                   w = rep(0, J),
                   z = rep(1, J))
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = list(0, 0, 0, 0),
                                       var = list(2.72, 2.72, 2.72, 2.72)),
                   phi.unif = c(3/1, 3/.1),
                   sigma.sq.ig = c(2, 2))
# Tuning
tuning.list <- list(phi = 1)

# Number of batches
n.batch <- 40
# Batch length
batch.length <- 25

out <- spIntPGOcc(occ.formula = ~ occ.cov,
                  det.formula = list(f.1 = ~ det.cov.1.1,
                                     f.2 = ~ det.cov.2.1 + det.cov.2.2,
                                     f.3 = ~ det.cov.3.1,
                                     f.4 = ~ det.cov.4.1 + det.cov.4.2 + det.cov.4.3),
                  data = data.list,
                  inits = inits.list,
                  n.batch = n.batch,
                  batch.length = batch.length,
                  accept.rate = 0.43,
                  priors = prior.list,
                  cov.model = "spherical",
                  tuning = tuning.list,
```

```
                       n.omp.threads = 1,
                       verbose = TRUE,
                       NNGP = TRUE,
                       n.neighbors = 5,
                       search.type = 'cb',
                       n.report = 10,
                       n.burn = 500,
                       n.thin = 1)
summary(out)

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

---

predict.spMsPGOcc   *Function for prediction at new locations for multi-species spatial oc-*
                                  cupancy models

Description

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

Usage

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

Arguments

object	an object of class spMsPGOcc
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 occupancy (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 spMsPGOcc. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of spMsPGOcc. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of spMsPGOcc.
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, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
...	currently no additional arguments

Value

An list object of class `predict.spMsPGOcc`. When `type = 'occupancy'`, the list consists of:

psi.0.samples	a three-dimensional array of posterior predictive samples for the latent occurrence probability values.
z.0.samples	a three-dimensional array of posterior predictive samples for the latent occurrence values.
w.0.samples	a three-dimensional array of posterior predictive samples for the latent spatial random effects.
run.time	execution time reported using `proc.time()`.

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

p.0.samples	a three-dimensional array of posterior predictive samples for the detection probability values.
run.time	execution time reported using `proc.time()`.

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(400)

# Simulate Data -------------------------------------------------------------
J.x <- 7
J.y <- 7
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
N <- 5
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -.2)
tau.sq.alpha <- c(0.2, 0.3, 0.8)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
phi <- runif(N, 3/1, 3/.4)
sigma.sq <- runif(N, 0.3, 3)
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
phi = phi, sigma.sq = sigma.sq, sp = TRUE, cov.model = 'exponential')

# Number of batches
n.batch <- 30
# Batch length
batch.length <- 25
n.samples <- n.batch * batch.length

# 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, ]
# Coordinates
coords <- as.matrix(dat$coords[-pred.indx, ])
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
coords.0 <- as.matrix(dat$coords[pred.indx, ])
```

```
psi.0 <- dat$psi[, pred.indx]

# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
 det.cov.2 = X.p[, , 3]
 )
data.list <- list(y = y,
  occ.covs = occ.covs,
  det.covs = det.covs,
  coords = coords)

# Priors
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, b = 3/.1),
   sigma.sq.ig = list(a = 2, b = 2))
# Starting values
inits.list <- list(alpha.comm = 0,
      beta.comm = 0,
      beta = 0,
      alpha = 0,
      tau.sq.beta = 1,
      tau.sq.alpha = 1,
      phi = 3 / .5,
      sigma.sq = 2,
      w = matrix(0, nrow = N, ncol = nrow(X)),
      z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- spMsPGOcc(occ.formula = ~ occ.cov,
 det.formula = ~ det.cov.1 + det.cov.2,
 data = data.list,
 inits = inits.list,
 n.batch = n.batch,
 batch.length = batch.length,
 accept.rate = 0.43,
         priors = prior.list,
 cov.model = "exponential",
 tuning = tuning.list,
         n.omp.threads = 1,
         verbose = TRUE,
 NNGP = TRUE,
 n.neighbors = 5,
 search.type = 'cb',
         n.report = 10,
 n.burn = 500,
 n.thin = 1)
```

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

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

---

predict.spPGOcc	*Function for prediction at new locations for single-species spatial occupancy models*

Description

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

Usage

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

Arguments

object	an object of class `spPGOcc`
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 occupancy (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 `spPGOcc`. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of `spPGOcc`. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of `spPGOcc`.
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.
ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.

n.report   the interval to report sampling progress.

type   a quoted keyword indicating what type of prediction to produce. Valid keywords
                are 'occupancy' to predict latent occupancy probability and latent occupancy
                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.spPGOcc`. When `type = 'occupancy'`, the list consists of:

psi.0.samples   a coda object of posterior predictive samples for the latent occurrence probabil-
                ity values.

z.0.samples   a coda object of posterior predictive samples for the latent occurrence values.

w.0.samples   a coda object of posterior predictive samples for the latent spatial random ef-
                fects.

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

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

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

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

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 sup-
ported for prediction. For sampled levels, the posterior distribution for the random intercept corre-
sponding 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>

References

Hooten, M. B., and Hefley, T. J. (2019). Bringing Bayesian models to life. CRC Press.

Examples

```
set.seed(400)
# Simulate Data -----------------------------------------------------------
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, 2)
p.occ <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
phi <- 3 / .6
sigma.sq <- 2
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sigma.sq = sigma.sq, phi = phi, sp = TRUE, cov.model = 'exponential')
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .5), 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, , ]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
psi.0 <- dat$psi[pred.indx]
w.0 <- dat$w[pred.indx]

# Package all data into a list
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = c(2, 2),
                   phi.unif = c(3/1, 3/.1))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
                   phi = 3 / .5,
                   sigma.sq = 2,
```

```
                          w = rep(0, nrow(X)),
                          z = apply(y, 1, max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- spPGOcc(occ.formula = ~ occ.cov,
               det.formula = ~ det.cov.1,
               data = data.list,
               inits = inits.list,
               n.batch = n.batch,
               batch.length = batch.length,
               accept.rate = 0.43,
               priors = prior.list,
               cov.model = 'exponential',
               tuning = tuning.list,
               n.omp.threads = 1,
               verbose = TRUE,
               NNGP = FALSE,
               n.neighbors = 15,
               search.type = 'cb',
               n.report = 10,
               n.burn = 50,
               n.thin = 1)

summary(out)

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

---

predict.stMsPGOcc	*Function for prediction at new locations for multi-season multi-species spatial occupancy models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'stMsPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'stMsPGOcc'
predict(object, X.0, coords.0, t.cols, n.omp.threads = 1,
                        verbose = TRUE, n.report = 100,
                        ignore.RE = FALSE, type = 'occupancy', ...)
```

Arguments

object   an object of class stMsPGOcc

X.0   the design matrix of covariates at the prediction locations. This should be a
                three-dimensional array, with dimensions corresponding to site, primary time
                period, 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 occupancy (or detection if type = 'detection') 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 stMsPGOcc. The
                covariates should be organized in the same order as they were specified in the
                corresponding formula argument of stMsPGOcc. 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 stMsPGOcc. See example below.

coords.0   the spatial coordinates corresponding to X.0. Note that spOccupancy assumes
                coordinates are specified in a projected coordinate system.

t.cols   an indexing vector used to denote which primary time periods are contained
                in the design matrix of covariates at the prediction locations (X.0). The values
                should denote the specific primary time periods used to fit the model. The values
                should indicate the columns in data$y used to fit the model for which prediction
                is desired. See example below.

n.omp.threads   a positive integer indicating the number of threads to use for SMP parallel pro-
                cessing. 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 sys-
                tems.

verbose   if TRUE, model specification and progress of the sampler is printed to the screen.
                Otherwise, nothing is printed to the screen.

ignore.RE   logical value that specifies whether or not to remove random unstructured occur-
                rence (or detection if type = 'detection') effects from the subsequent predic-
                tions. If TRUE, random effects will be included. If FALSE, unstructured random
                effects will be set to 0 and predictions will only be generated from the fixed ef-
                fects, the spatial random effects, and AR(1) random effects if the model was fit
                with ar1 = TRUE.

n.report   the interval to report sampling progress.

type   a quoted keyword indicating what type of prediction to produce. Valid keywords
                are 'occupancy' to predict latent occupancy probability and latent occupancy
                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.stMsPGOcc. When type = 'occupancy', the list consists of:

psi.0.samples   a four-dimensional object of posterior predictive samples for the latent occu-
                pancy probability values with dimensions corresponding to posterior predictive
                sample, species, site, and primary time period.

z.0.samples   a three-dimensional object of posterior predictive samples for the latent occu-
                 pancy values with dimensions corresponding to posterior predictive sample,
                 species, site, and primary time period.

w.0.samples   a three-dimensional array of posterior predictive samples for the latent spatial
                 factors with dimensions correpsonding to MCMC sample, latent factor, and site.

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

p.0.samples   a four-dimensional object of posterior predictive samples for the detection prob-
                 ability values with dimensions corresponding to posterior predictive sample,
                 site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary
time periods are obtained directly when fitting the model. See the psi.samples and z.samples
portions of the output list from the model object of class stMsPGOcc.

Author(s)

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

Examples

```
# Simulate Data ----------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
  # n.rep[j, 1:n.time[j]] <- rep(4, n.time[j])
}
N <- 7
# Community-level covariate effects
# Occurrence
```

```
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
svc.cols <- c(1)
p.svc <- length(svc.cols)
n.factors <- 3
phi <- runif(p.svc * n.factors, 3 / .9, 3 / .3)
factor.model <- TRUE
cov.model <- 'exponential'
ar1 <- TRUE
sigma.sq.t <- runif(N, 0.05, 1)
rho <- runif(N, 0.1, 1)

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
 psi.RE = psi.RE, p.RE = p.RE, factor.model = factor.model,
                 svc.cols = svc.cols, n.factors = n.factors, phi = phi, sp = sp,
                 cov.model = cov.model, ar1 = ar1, sigma.sq.t = sigma.sq.t, rho = rho)

# Subset data for prediction
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 covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]
# Coordinates
coords <- dat$coords[-pred.indx, ]
coords.0 <- dat$coords[pred.indx, ]

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

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

data.list <- list(y = y, occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
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),
   rho.unif = list(a = -1, b = 1),
   sigma.sq.t.ig = list(a = 0.1, b = 0.1),
                  phi.unif = list(a = 3 / .9, b = 3 / .1))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
   alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
   rho = 0.5, sigma.sq.t = 0.5,
   phi = 3 / .5, z = z.init)
# Tuning
tuning.list <- list(phi = 1, rho = 0.5)

# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

out <- stMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 accept.rate = 0.43,
ar1 = TRUE,
NNGP = TRUE,
n.neighbors = 5,
n.factors = n.factors,
cov.model = 'exponential',
                 priors = prior.list,
                 tuning = tuning.list,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 n.report = 1,
                 n.burn = n.burn,
n.thin = n.thin,
n.chains = 1)

summary(out)
```

```
# Predict at new sites across all n.max.years
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.pred, coords.0, t.cols = t.cols, type = 'occupancy')
str(out.pred)
```

predict.stPGOcc	*Function for prediction at new locations for multi-season single-species spatial occupancy models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'stPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'stPGOcc'
predict(object, X.0, coords.0, t.cols, n.omp.threads = 1,
                         verbose = TRUE, n.report = 100,
                         ignore.RE = FALSE, type = 'occupancy',
                         forecast = FALSE, ...)
```

Arguments

object	an object of class stPGOcc
X.0	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, primary time period, 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 occupancy (or detection if `type = 'detection'`) 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 `stPGOcc`. The covariates should be organized in the same order as they were specified in the corresponding formula argument of `stPGOcc`. 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 `stPGOcc`. See example below.
coords.0	the spatial coordinates corresponding to `X.0`. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

t.cols	an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (X.0). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in data$y used to fit the model for which prediction is desired. See example below. Not required when forecast = TRUE.
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.
ignore.RE	logical value that specifies whether or not to remove random unstructured occurrence (or detection if type = 'detection') effects from the subsequent predictions. If TRUE, 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, the spatial random effects, and AR(1) random effects if the model was fit with ar1 = TRUE.
n.report	the interval to report sampling progress.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy values (this is the default), or 'detection' to predict detection probability given new values of detection covariates.
forecast	a logical value indicating whether prediction is occurring at non-sampled primary time periods (e.g., forecasting).
...	currently no additional arguments

Value

A list object of class predict.stPGOcc. When type = 'occupancy', the list consists of:

psi.0.samples	a three-dimensional object of posterior predictive samples for the latent occupancy probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.
z.0.samples	a three-dimensional object of posterior predictive samples for the latent occupancy values with dimensions corresponding to posterior predictive sample, site, and primary time period.
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 three-dimensional object of posterior predictive samples for the detection probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary time periods are obtained directly when fitting the model. See the psi.samples and z.samples portions of the output list from the model object of class stPGOcc.

Author(s)

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

Examples

```
set.seed(500)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Primary time periods
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence -------------------------
beta <- c(0.4, 0.5, -0.9)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection --------------------------
alpha <- c(-1, 0.7, -0.5)
p.RE <- list()
# Spatial ----------------------------
sp <- TRUE
cov.model <- "exponential"
sigma.sq <- 2
phi <- 3 / .4

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
               beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
               psi.RE = psi.RE, p.RE = p.RE, sp = TRUE, sigma.sq = sigma.sq,
               phi = phi, cov.model = cov.model, ar1 = FALSE)
```

```
# Subset data for prediction
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 covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]
psi.0 <- dat$psi[pred.indx, ]
# Coordinates
coords <- dat$coords[-pred.indx, ]
coords.0 <- dat$coords[pred.indx, ]

# Package all data into a list
# Occurrence
occ.covs <- list(int = X[, , 1],
                 trend = X[, , 2],
                 occ.cov.1 = X[, , 3])
# Detection
det.covs <- list(det.cov.1 = X.p[, , , 2],
                 det.cov.2 = X.p[, , , 3])
# Data list bundle
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = c(2, 2),
                   phi.unif = c(3 / 1, 3 / 0.1))

# Initial values
z.init <- apply(y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0, z = z.init, phi = 3 / .5, sigma.sq = 2,
                   w = rep(0, J))
# Tuning
tuning.list <- list(phi = 1)
# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length

# Run the model
out <- stPGOcc(occ.formula = ~ trend + occ.cov.1,
               det.formula = ~ det.cov.1 + det.cov.2,
               data = data.list,
               inits = inits.list,
               n.batch = n.batch,
               batch.length = batch.length,
               priors = prior.list,
```

```
                    cov.model = "exponential",
                    tuning = tuning.list,
                    NNGP = TRUE,
                    ar1 = FALSE,
                    n.neighbors = 5,
                    search.type = 'cb',
                    n.report = 10,
                    n.burn = 50,
                    n.chains = 1)

summary(out)

# Predict at new sites across all n.max.years
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.0, coords.0, t.cols = t.cols, type = 'occupancy')
str(out.pred)
```

---

predict.svcMsPGOcc	*Function for prediction at new locations for spatially varying coeffi-cient multi-species occupancy models*

Description

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

Usage

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

Arguments

object	an object of class svcMsPGOcc
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 occupancy (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 svcMsPGOcc. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of svcMsPGOcc. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of svcMsPGOcc.

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 pro-
                 cessing. 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 sys-
                 tems.

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, unstructured random effects will be ignored and prediction
                 will only use the fixed effects and the spatial random effects. If FALSE, random
                 effects will be included in the prediction for both observed and unobserved levels
                 of the unstructured random effects.

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

...   currently no additional arguments

Value

An list object of class predict.svcMsPGOcc. When type = 'occupancy', the list consists of:

psi.0.samples   a three-dimensional array of posterior predictive samples for the latent occur-
                 rence probability values.

z.0.samples   a three-dimensional array of posterior predictive samples for the latent occur-
                 rence values.

w.0.samples   a four-dimensional array of posterior predictive samples for the spatially-varying
                 coefficients, with dimensions corresponding to MCMC sample, spatial factor,
                 site, and spatially varying coefficient.

run.time   execution time reported using proc.time().

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

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

run.time   execution time reported using proc.time().

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(400)

# Simulate Data -----------------------------------------------------------
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- sample(5, size = J, replace = TRUE)
N <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.2, 0.3, -0.1, 0.4)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 0.4, 0.5, 0.3)
# Detection
alpha.mean <- c(0, 1.2, -0.5)
tau.sq.alpha <- c(1, 0.5, 1.3)
p.det <- length(alpha.mean)
# No random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
# Number of spatial factors for each SVC
n.factors <- 2
# The intercept and first two covariates have spatially-varying effects
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
q.p.svc <- n.factors * p.svc
# Spatial decay parameters
```

```
phi <- runif(q.p.svc, 3 / 0.9, 3 / 0.1)
# A length N vector indicating the proportion of simulated locations
# that are within the range for a given species.
range.probs <- runif(N, 1, 1)
factor.model <- TRUE
cov.model <- 'spherical'
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
psi.RE = psi.RE, p.RE = p.RE, phi = phi, sp = sp, svc.cols = svc.cols,
cov.model = cov.model, n.factors = n.factors,
factor.model = factor.model, range.probs = range.probs)

# 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, ]
# Coordinates
coords <- as.matrix(dat$coords[-pred.indx, ])
# Detection covariates
X.p <- dat$X.p[-pred.indx, , ]
# Prediction values
X.0 <- dat$X[pred.indx, ]
coords.0 <- as.matrix(dat$coords[pred.indx, ])

# Prep data for spOccupancy ----------------------------------------------
# Occurrence covariates
occ.covs <- cbind(X)
colnames(occ.covs) <- c('int', 'occ.cov.1', 'occ.cov.2', 'occ.cov.3',
'occ.cov.4')
# Detection covariates
det.covs <- list(det.cov.1 = X.p[, , 2],
 det.cov.2 = X.p[, , 3])
# Data list
data.list <- list(y = y, coords = coords, occ.covs = occ.covs,
                  det.covs = det.covs)
# Priors
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, b = 3 / .1))
inits.list <- list(alpha.comm = 0,
   beta.comm = 0,
   beta = 0,
   alpha = 0,
   tau.sq.beta = 1,
   tau.sq.alpha = 1,
   z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)
```

```
# Number of batches
n.batch <- 2
# Batch length
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.samples <- n.batch * batch.length

out <- svcMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2 + occ.cov.3 +
                                  occ.cov.4,
                  det.formula = ~ det.cov.1 + det.cov.2,
                  data = data.list,
                  inits = inits.list,
                  n.batch = n.batch,
   n.factors = n.factors,
                  batch.length = batch.length,
  std.by.sp = TRUE,
                  accept.rate = 0.43,
                  priors = prior.list,
  svc.cols = svc.cols,
                  cov.model = "spherical",
                  tuning = tuning.list,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  NNGP = TRUE,
                  n.neighbors = 5,
                  search.type = 'cb',
                  n.report = 10,
                  n.burn = n.burn,
  n.thin = n.thin,
  n.chains = 1)

summary(out)
# Predict at new locations ------------------------------------------
out.pred <- predict(out, X.0, coords.0, verbose = FALSE)

# Get SVC samples for each species at prediction locations
svc.samples <- getSVCSamples(out, out.pred)
```

---

predict.svcPGBinom   *Function for prediction at new locations for single-species spatially-varying coefficient Binomial models*

Description

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

Usage

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

Arguments

object	an object of class svcPGBinom
X.0	the design matrix of covariates at the prediction locations. Note that for spatially-varying coefficients models the order of covariates in X.0 must be the same as the order of covariates specified in the model formula. This should include a column of 1s for the intercept if an intercept is included in the model. If unstructured 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 svcPGBinom. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of svcPGBinom. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of svcPGBinom.
coords.0	the spatial coordinates corresponding to X.0. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.
weights.0	a numeric vector containing the binomial weights (i.e., the total number of Bernoulli trials) at each site. If weights.0 is not specified, we assume 1 trial at each site (i.e., presence/absence).
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.
ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.
n.report	the interval to report sampling progress.
...	currently no additional arguments

Value

A list object of class predict.svcPGBinom consisting of:

psi.0.samples	a coda object of posterior predictive samples for the binomial probability values.
y.0.samples	a coda object of posterior predictive samples for the binomial data.

w.0.samples   a three-dimensional array of posterior predictive samples for the spatial random effects, with dimensions corresponding to MCMC iteration, coefficient, and site.

run.time   execution time reported using proc.time().

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
# Binomial weights
weights <- sample(10, J, replace = TRUE)
beta <- c(0, 0.5, -0.2, 0.75)
p <- length(beta)
# No unstructured random effects
psi.RE <- list()
# Spatial parameters
sp <- TRUE
# Two spatially-varying covariates.
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.4, 1.5)
phi <- runif(p.svc, 3/1, 3/0.2)

# Simulate the data
dat <- simBinom(J.x = J.x, J.y = J.y, weights = weights, beta = beta,
                psi.RE = psi.RE, sp = sp, svc.cols = svc.cols,
                cov.model = cov.model, sigma.sq = sigma.sq, phi = phi)

# Binomial data
y <- dat$y
# Covariates
X <- dat$X
# Spatial coordinates
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, ]
weights.0 <- weights[pred.indx]
weights <- weights[-pred.indx]

# Package all data into a list
# Covariates
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,
                  weights = weights)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   sigma.sq.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)
# Tuning
tuning.list <- list(phi = 1)

n.batch <- 10
batch.length <- 25
n.burn <- 100
n.thin <- 1

out <- svcPGBinom(formula = ~ cov.1 + cov.2 + cov.3,
                  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,
                  cov.model = "exponential",
                  tuning = tuning.list,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  NNGP = TRUE,
                  n.neighbors = 5,
                  n.report = 2,
```

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

summary(out)

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

---

predict.svcPGOcc	*Function for prediction at new locations for single-species spatially-varying coefficient occupancy models*

Description

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

Usage

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

Arguments

object	an object of class svcPGOcc
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 occupancy (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 svcPGOcc. Columns should correspond to the order of how covariates were specified in the corresponding formula argument of svcPGOcc. Column names of the random effects must match the name of the random effects, if specified in the corresponding formula argument of svcPGOcc.
coords.0	the spatial coordinates corresponding to `X.0`. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.
weights.0	not used for objects of class svcTPGOcc. Used when calling other functions.
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.
ignore.RE	a logical value indicating whether to include unstructured random effects for prediction. If TRUE, unstructured random effects will be ignored and prediction will only use the fixed effects and the spatial random effects. If FALSE, random effects will be included in the prediction for both observed and unobserved levels of the unstructured random effects.
n.report	the interval to report sampling progress.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy 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.svcPGOcc`. When `type = 'occupancy'`, the list consists of:

psi.0.samples	a coda object of posterior predictive samples for the latent occurrence probability values.
z.0.samples	a coda object of posterior predictive samples for the latent occurrence values.
w.0.samples	a three-dimensional array of posterior predictive samples for the spatial random effects, with dimensions corresponding to MCMC iteration, coefficient, and site.
run.time	execution time reported using `proc.time()`.

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

p.0.samples	a coda object of posterior predictive samples for the detection probability values.
run.time	execution time reported using `proc.time()`.

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>

References

Hooten, M. B., and Hefley, T. J. (2019). Bringing Bayesian models to life. CRC Press.

Examples

```
set.seed(400)
# Simulate Data -----------------------------------------------------------
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, 2)
p.occ <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
phi <- c(3 / .6, 3 / .8)
sigma.sq <- c(0.5, 0.9)
svc.cols <- c(1, 2)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sigma.sq = sigma.sq, phi = phi, sp = TRUE, cov.model = 'exponential',
              svc.cols = svc.cols)
# Split into fitting and prediction data set
pred.indx <- sample(1:J, round(J * .5), 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, , ]
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
psi.0 <- dat$psi[pred.indx]
w.0 <- dat$w[pred.indx, , drop = FALSE]

# Package all data into a list
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = list(a = 2, b = 0.5),
                   phi.unif = list(a = 3/1, b = 3/.1))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
```

```
                            phi = 3 / .5,
                            sigma.sq = 0.5,
                            z = apply(y, 1, max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- svcPGOcc(occ.formula = ~ occ.cov,
                det.formula = ~ det.cov.1,
                data = data.list,
                inits = inits.list,
                n.batch = n.batch,
                batch.length = batch.length,
                accept.rate = 0.43,
                priors = prior.list,
                cov.model = 'exponential',
                tuning = tuning.list,
                n.omp.threads = 1,
                verbose = TRUE,
                NNGP = TRUE,
                svc.cols = c(1, 2),
                n.neighbors = 15,
                search.type = 'cb',
                n.report = 10,
                n.burn = 50,
                n.thin = 1)

summary(out)

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

---

predict.svcTMsPGOcc	*Function for prediction at new locations for multi-season multi-species spatially-varying coefficient occupancy models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'svcTMsPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'svcTMsPGOcc'
predict(object, X.0, coords.0, t.cols, n.omp.threads = 1,
                        verbose = TRUE, n.report = 100,
                        ignore.RE = FALSE, type = 'occupancy', ...)
```

Arguments

`object`	an object of class svcTMsPGOcc
`X.0`	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, primary time period, 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 occupancy (or detection if `type = 'detection'`) 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 `svcTMsPGOcc`. The covariates should be organized in the same order as they were specified in the corresponding formula argument of `svcTMsPGOcc`. 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 `svcTMsPGOcc`. See example below.
`coords.0`	the spatial coordinates corresponding to `X.0`. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.
`t.cols`	an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (`X.0`). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in data$y used to fit the model for which prediction is desired. See example below.
`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.
`ignore.RE`	logical value that specifies whether or not to remove random unstructured occurrence (or detection if `type = 'detection'`) effects from the subsequent predictions. If TRUE, 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, the spatial random effects, and AR(1) random effects if the model was fit with `ar1 = TRUE`.
`n.report`	the interval to report sampling progress.
`type`	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy 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.svcTMsPGOcc`. When `type = 'occupancy'`, the list consists of:

psi.0.samples   a four-dimensional object of posterior predictive samples for the latent occu-
                 pancy probability values with dimensions corresponding to posterior predictive
                 sample, species, site, and primary time period.

z.0.samples   a three-dimensional object of posterior predictive samples for the latent occu-
                 pancy values with dimensions corresponding to posterior predictive sample,
                 species, site, and primary time period.

w.0.samples   a four-dimensional array of posterior predictive samples for the latent spatial
                 factors with dimensions correpsonding to MCMC sample, latent factor, site, and
                 spatially-varying coefficient.

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

p.0.samples   a four-dimensional object of posterior predictive samples for the detection prob-
                 ability values with dimensions corresponding to posterior predictive sample,
                 site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary
time periods are obtained directly when fitting the model. See the psi.samples and z.samples
portions of the output list from the model object of class svcTMsPGOcc.

Author(s)

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

Examples

```
# Simulate Data -------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
```

```
    # n.rep[j, 1:n.time[j]] <- rep(4, n.time[j])
  }
  N <- 7
  # Community-level covariate effects
  # Occurrence
  beta.mean <- c(-3, -0.2, 0.5)
  trend <- FALSE
  sp.only <- 0
  p.occ <- length(beta.mean)
  tau.sq.beta <- c(0.6, 1.5, 1.4)
  # Detection
  alpha.mean <- c(0, 1.2, -1.5)
  tau.sq.alpha <- c(1, 0.5, 2.3)
  p.det <- length(alpha.mean)
  # Random effects
  psi.RE <- list()
  p.RE <- list()
  # Draw species-level effects from community means.
  beta <- matrix(NA, nrow = N, ncol = p.occ)
  alpha <- matrix(NA, nrow = N, ncol = p.det)
  for (i in 1:p.occ) {
    beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
  }
  for (i in 1:p.det) {
    alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
  }
  sp <- TRUE
  svc.cols <- c(1, 2)
  p.svc <- length(svc.cols)
  n.factors <- 2
  phi <- runif(p.svc * n.factors, 3 / .9, 3 / .3)
  factor.model <- TRUE
  cov.model <- 'exponential'
  ar1 <- TRUE
  sigma.sq.t <- runif(N, 0.05, 1)
  rho <- runif(N, 0.1, 1)

  dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
   beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
   psi.RE = psi.RE, p.RE = p.RE, factor.model = factor.model,
                   svc.cols = svc.cols, n.factors = n.factors, phi = phi, sp = sp,
                   cov.model = cov.model, ar1 = ar1, sigma.sq.t = sigma.sq.t, rho = rho)

  # Subset data for prediction
  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 covariates
  X.0 <- dat$X[pred.indx, , , drop = FALSE]
  # Detection covariates
  X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]
  # Coordinates
```

```
coords <- dat$coords[-pred.indx, ]
coords.0 <- dat$coords[pred.indx, ]

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

data.list <- list(y = y, occ.covs = occ.covs,
                   det.covs = det.covs,
                   coords = coords)
# Priors
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),
   rho.unif = list(a = -1, b = 1),
   sigma.sq.t.ig = list(a = 0.1, b = 0.1),
                    phi.unif = list(a = 3 / .9, b = 3 / .1))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
   alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
   rho = 0.5, sigma.sq.t = 0.5,
   phi = 3 / .5, z = z.init)
# Tuning
tuning.list <- list(phi = 1, rho = 0.5)

# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

out <- svcTMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 accept.rate = 0.43,
ar1 = TRUE,
svc.cols = svc.cols,
NNGP = TRUE,
n.neighbors = 5,
n.factors = n.factors,
cov.model = 'exponential',
                 priors = prior.list,
                 tuning = tuning.list,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 n.report = 1,
```

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

summary(out)

# Predict at new sites across all n.max.years
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.pred, coords.0, t.cols = t.cols, type = 'occupancy')
str(out.pred)

# Extract SVC samples for each species at prediction locations
svc.samples <- getSVCSamples(out, out.pred)
str(svc.samples)
```

---

predict.svcTPGBinom	*Function for prediction at new locations for multi-season single-species spatially-varying coefficient binomial models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'svcTPGBinom'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'svcTPGBinom'
predict(object, X.0, coords.0, t.cols, weights.0,  n.omp.threads = 1,
        verbose = TRUE, n.report = 100, ignore.RE = FALSE, ...)
```

Arguments

object   an object of class svcTPGBinom

X.0   the design matrix of covariates at the prediction locations. This should be a
              three-dimensional array, with dimensions corresponding to site, primary time
              period, 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 occupancy (or detection if `type = 'detection'`) 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 `svcTPGBinom`.
              The covariates should be organized in the same order as they were specified
              in the corresponding formula argument of `svcTPGBinom`. Names of the third

<table>
<tr><td></td><td>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 svcTPGBinom. See example below.</td></tr>
</table>

coords.0	the spatial coordinates corresponding to X.0. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.
weights.0	a numeric site by primary time period matrix containing the binomial weights (i.e., the total number of Bernoulli trials) at each site and primary time period. If weights.0 is not specified, we assume 1 trial at each site/primary time period (i.e., presence/absence).
t.cols	an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (X.0). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in data$y used to fit the model for which prediction is desired. See example below.
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.
ignore.RE	logical value that specifies whether or not to remove random unstructured occurrence (or detection if type = 'detection') effects from the subsequent predictions. If TRUE, 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, the spatial random effects, and AR(1) random effects if the model was fit with ar1 = TRUE.
n.report	the interval to report sampling progress.
...	currently no additional arguments

Value

A list object of class predict.svcTPGBinom that consists of:

psi.0.samples	a three-dimensional object of posterior predictive samples for the occurrence probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.
y.0.samples	a three-dimensional object of posterior predictive samples for the predicted binomial data with dimensions corresponding to posterior predictive sample, site, and primary time period.
w.0.samples	a three-dimensional array of posterior predictive samples for the spatial random effects, with dimensions corresponding to MCMC iteration, coefficient, and site.
run.time	execution time reported using proc.time().

Note

When `ignore.RE = FALSE`, both sampled levels and non-sampled levels of unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary time periods are obtained directly when fitting the model. See the `psi.samples` and `y.rep.samples` portions of the output list from the model object of class `svcTPGBinom`.

Author(s)

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

Examples

```
set.seed(1000)
# Sites
J.x <- 15
J.y <- 15
J <- J.x * J.y
# Years sampled
n.time <- sample(10, J, replace = TRUE)
# Binomial weights
weights <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  weights[j, 1:n.time[j]] <- sample(5, n.time[j], replace = TRUE)
}
# Occurrence ------------------------
beta <- c(-2, -0.5, -0.2, 0.75)
p.occ <- length(beta)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Spatial parameters -----------------
sp <- TRUE
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.1, 1)
phi <- runif(p.svc, 3/1, 3/0.2)
# Temporal parameters ----------------
ar1 <- TRUE
rho <- 0.8
sigma.sq.t <- 1

# Get all the data
dat <- simTBinom(J.x = J.x, J.y = J.y, n.time = n.time, weights = weights, beta = beta,
                 psi.RE = psi.RE, sp.only = sp.only, trend = trend,
```

```
                          sp = sp, svc.cols = svc.cols,
                          cov.model = cov.model, sigma.sq = sigma.sq, phi = phi,
                          rho = rho, sigma.sq.t = sigma.sq.t, ar1 = TRUE, x.positive = FALSE)

# Prep the data for spOccupancy -----------------------------------------
# Subset data for prediction
pred.indx <- sample(1:J, round(J * .25), replace = FALSE)
y <- dat$y[-pred.indx, , drop = FALSE]
y.0 <- dat$y[pred.indx, , drop = FALSE]
# Occupancy covariates
X <- dat$X[-pred.indx, , , drop = FALSE]
# Prediction covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Spatial coordinates
coords <- as.matrix(dat$coords[-pred.indx, ])
coords.0 <- as.matrix(dat$coords[pred.indx, ])
psi.0 <- dat$psi[pred.indx, ]
w.0 <- dat$w[pred.indx, ]
weights.0 <- weights[pred.indx, ]
weights <- weights[-pred.indx, ]

# Package all data into a list
covs <- list(int = X[, , 1],
             trend = X[, , 2],
             cov.1 = X[, , 3],
             cov.2 = X[, , 4])
# Data list bundle
data.list <- list(y = y,
                  covs = covs,
                  weights = weights,
                  coords = coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = list(a = 2, b = 1),
                   phi.unif = list(a = 3/1, b = 3/.1))

# Starting values
inits.list <- list(beta = beta, alpha = 0,
                   sigma.sq = 1, phi = 3 / 0.5, nu = 1)
# Tuning
tuning.list <- list(phi = 0.4, nu = 0.3, rho = 0.2)

# MCMC information
n.batch <- 2
n.burn <- 0
n.thin <- 1


out <- svcTPGBinom(formula = ~ trend + cov.1 + cov.2,
                   svc.cols = svc.cols,
                   data = data.list,
                   n.batch = n.batch,
                   batch.length = 25,
```

```
                        inits = inits.list,
                        priors = prior.list,
                        accept.rate = 0.43,
                        cov.model = "exponential",
                        ar1 = TRUE,
                        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 = 1)
# Predict at new locations ------------------------------------------
out.pred <- predict(out, X.0, coords.0, t.cols = 1:max(n.time),
                    weights = weights.0, n.report = 10)
str(out.pred)
```

---

predict.svcTPGOcc   *Function for prediction at new locations for multi-season single-species spatially-varying coefficient occupancy models*

Description

The function `predict` collects posterior predictive samples for a set of new locations given an object of class 'svcTPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'svcTPGOcc'
predict(object, X.0, coords.0, t.cols, weights.0, n.omp.threads = 1,
        verbose = TRUE, n.report = 100,
        ignore.RE = FALSE, type = 'occupancy', forecast = FALSE, ...)
```

Arguments

object	an object of class svcTPGOcc
X.0	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, primary time period, 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 occupancy (or detection if `type = 'detection'`) 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 `svcTPGOcc`. The covariates should be organized in the same order as they were specified in the

corresponding formula argument of `svcTPGOcc`. 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 `svcTPGOcc`. See example below.

coords.0   the spatial coordinates corresponding to X.0. Note that `spOccupancy` assumes coordinates are specified in a projected coordinate system.

t.cols   an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (X.0). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in `data$y` used to fit the model for which prediction is desired. See example below. Not required when `forecast = TRUE`.

weights.0   not used for objects of class `svcTPGOcc`. Used when calling other functions.

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.

ignore.RE   logical value that specifies whether or not to remove random unstructured occurrence (or detection if `type = 'detection'`) effects from the subsequent predictions. If TRUE, 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, the spatial random effects, and AR(1) random effects if the model was fit with `ar1 = TRUE`.

n.report   the interval to report sampling progress.

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

...   currently no additional arguments

forecast   a logical value indicating whether prediction is occurring at non-sampled primary time periods (e.g., forecasting).

Value

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

psi.0.samples   a three-dimensional object of posterior predictive samples for the latent occupancy probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.

z.0.samples   a three-dimensional object of posterior predictive samples for the latent occupancy values with dimensions corresponding to posterior predictive sample, site, and primary time period.

w.0.samples   a three-dimensional array of posterior predictive samples for the spatial random effects, with dimensions corresponding to MCMC iteration, coefficient, and site.

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

p.0.samples   a three-dimensional object of posterior predictive samples for the detection probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary time periods are obtained directly when fitting the model. See the `psi.samples` and `z.samples` portions of the output list from the model object of class `svcTPGOcc`.

Author(s)

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

Examples

```
set.seed(500)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Primary time periods
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence -------------------------
beta <- c(0.4, 0.5, -0.9)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection --------------------------
alpha <- c(-1, 0.7, -0.5)
p.RE <- list()
# Spatial ----------------------------
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
```

```
sp <- TRUE
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.1, 1)
phi <- runif(p.svc, 3 / .9, 3 / .1)

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
                beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
                psi.RE = psi.RE, p.RE = p.RE, sp = TRUE, sigma.sq = sigma.sq,
                phi = phi, cov.model = cov.model, ar1 = FALSE, svc.cols = svc.cols)

# Subset data for prediction
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 covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]
psi.0 <- dat$psi[pred.indx, ]
# Coordinates
coords <- dat$coords[-pred.indx, ]
coords.0 <- dat$coords[pred.indx, ]

# Package all data into a list
# Occurrence
occ.covs <- list(int = X[, , 1],
                 trend = X[, , 2],
                 occ.cov.1 = X[, , 3])
# Detection
det.covs <- list(det.cov.1 = X.p[, , , 2],
                 det.cov.2 = X.p[, , , 3])
# Data list bundle
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = list(a = 2, b = 0.5),
                   phi.unif = list(a = 3 / 1, b = 3 / 0.1))

# Initial values
z.init <- apply(y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0, z = z.init, phi = 3 / .5, sigma.sq = 2,
                   w = rep(0, J))
# Tuning
tuning.list <- list(phi = 1)
# Number of batches
n.batch <- 10
# Batch length
```

```
batch.length <- 25
n.iter <- n.batch * batch.length

# Run the model
out <- svcTPGOcc(occ.formula = ~ trend + occ.cov.1,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 priors = prior.list,
                 cov.model = "exponential",
                 svc.cols = svc.cols,
                 tuning = tuning.list,
                 NNGP = TRUE,
                 ar1 = FALSE,
                 n.neighbors = 5,
                 search.type = 'cb',
                 n.report = 10,
                 n.burn = 50,
                 n.chains = 1)

summary(out)

# Predict at new sites across all n.max.years
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.0, coords.0, t.cols = t.cols, type = 'occupancy')
str(out.pred)
```

---

predict.tMsPGOcc	*Function for prediction at new locations for multi-season multi-species occupancy models*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'tMsPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'tMsPGOcc'
predict(object, X.0, t.cols, ignore.RE = FALSE, type = 'occupancy', ...)
```

Arguments

object	an object of class tMsPGOcc
X.0	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, primary time period, 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 occupancy (or detection if type = 'detection') 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 tMsPGOcc. The covariates should be organized in the same order as they were specified in the corresponding formula argument of tMsPGOcc. 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 tMsPGOcc. See example below.
t.cols	an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (X.0). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in data$y used to fit the model for which prediction is desired. See example below.
ignore.RE	logical value that specifies whether or not to remove random unstructured occurrence (or detection if type = 'detection') 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 and AR(1) random effects if the model was fit with ar1 = TRUE.
type	a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy 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.tMsPGOcc. When type = 'occupancy', the list consists of:

psi.0.samples	a four-dimensional object of posterior predictive samples for the latent occupancy probability values with dimensions corresponding to posterior predictive sample, species, site, and primary time period.
z.0.samples	a four-dimensional object of posterior predictive samples for the latent occupancy values with dimensions corresponding to posterior predictive sample, species, site, and primary time period.

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

p.0.samples	a four-dimensional object of posterior predictive samples for the detection probability values with dimensions corresponding to posterior predictive sample, species, site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary time periods are obtained directly when fitting the model. See the `psi.samples` and `z.samples` portions of the output list from the model object of class `tMsPGOcc`.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>

Examples

```
# Simulate Data -------------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
  # n.rep[j, 1:n.time[j]] <- rep(4, n.time[j])
}
N <- 7
# Community-level covariate effects
# Occurrence
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
```

```
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
 psi.RE = psi.RE, p.RE = p.RE, sp = sp)

# Subset data for prediction
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 covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]

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

data.list <- list(y = y, occ.covs = occ.covs,
                  det.covs = det.covs)
# Priors
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))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
   alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
   z = z.init)
# Tuning
tuning.list <- list(phi = 1)

# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

out <- tMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2,
                det.formula = ~ det.cov.1 + det.cov.2,
```

```
                data = data.list,
                inits = inits.list,
                n.batch = n.batch,
                batch.length = batch.length,
                accept.rate = 0.43,
                priors = prior.list,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1,
                n.burn = n.burn,
n.thin = n.thin,
n.chains = 1)

summary(out)

# Predict at new sites during time periods 1, 2, and 5
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.pred, t.cols = t.cols, type = 'occupancy')
str(out.pred)
```

---

predict.tPGOcc	*Function for prediction at new locations for multi-season single-species occupancy models*

Description

The function predict collects posterior predictive samples for a set of new locations given an object of class 'tPGOcc'. Prediction is possible for both the latent occupancy state as well as detection. Predictions are currently only possible for sampled primary time periods.

Usage

```
## S3 method for class 'tPGOcc'
predict(object, X.0, t.cols, ignore.RE = FALSE, type = 'occupancy', ...)
```

Arguments

object	an object of class tPGOcc
X.0	the design matrix of covariates at the prediction locations. This should be a three-dimensional array, with dimensions corresponding to site, primary time period, 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 occupancy (or detection if type = 'detection') 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 `tPGOcc`. The covariates should be organized in the same order as they were specified in the corresponding formula argument of `tPGOcc`. 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 `tPGOcc`. See example below.

`t.cols`   an indexing vector used to denote which primary time periods are contained in the design matrix of covariates at the prediction locations (`X.0`). The values should denote the specific primary time periods used to fit the model. The values should indicate the columns in `data$y` used to fit the model for which prediction is desired. See example below.

`ignore.RE`   logical value that specifies whether or not to remove random unstructured occurrence (or detection if `type = 'detection'`) 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 and AR(1) random effects if the model was fit with `ar1 = TRUE`.

`type`   a quoted keyword indicating what type of prediction to produce. Valid keywords are 'occupancy' to predict latent occupancy probability and latent occupancy 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.tPGOcc`. When `type = 'occupancy'`, the list consists of:

`psi.0.samples`   a three-dimensional object of posterior predictive samples for the latent occupancy probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.

`z.0.samples`   a three-dimensional object of posterior predictive samples for the latent occupancy values with dimensions corresponding to posterior predictive sample, site, and primary time period.

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

`p.0.samples`   a three-dimensional object of posterior predictive samples for the detection probability values with dimensions corresponding to posterior predictive sample, site, and primary time period.

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 unstructured 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.

Occurrence predictions at sites that are only sampled for a subset of the total number of primary time periods are obtained directly when fitting the model. See the `psi.samples` and `z.samples` portions of the output list from the model object of class `tPGOcc`.

Author(s)

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

Examples

```
set.seed(990)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Primary time periods
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence ------------------------
beta <- c(0.4, 0.5, -0.9)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection -------------------------
alpha <- c(-1, 0.7, -0.5)
p.RE <- list()

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
               beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
               psi.RE = psi.RE, p.RE = p.RE, sp = FALSE, ar1 = FALSE)

# Subset data for prediction
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 covariates
X.0 <- dat$X[pred.indx, , , drop = FALSE]
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , , drop = FALSE]
psi.0 <- dat$psi[pred.indx, ]

# Package all data into a list
```

```
# Occurrence
occ.covs <- list(int = X[, , 1],
                 trend = X[, , 2],
                 occ.cov.1 = X[, , 3])
# Detection
det.covs <- list(det.cov.1 = X.p[, , , 2],
                 det.cov.2 = X.p[, , , 3])
# Data list bundle
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72))

# Starting values
z.init <- apply(y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0, z = z.init)

n.batch <- 100
batch.length <- 25
n.burn <- 2000
n.thin <- 1

# Run the model
out <- tPGOcc(occ.formula = ~ trend + occ.cov.1,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              inits = inits.list,
              priors = prior.list,
              n.batch = n.batch,
              batch.length = batch.length,
              ar1 = FALSE,
              verbose = TRUE,
              n.report = 500,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = 1)

# Predict at new sites across during time periods 1, 2, and 5
# Take a look at array of covariates for prediction
str(X.0)
# Subset to only grab time periods 1, 2, and 5
t.cols <- c(1, 2, 5)
X.pred <- X.0[, t.cols, ]
out.pred <- predict(out, X.pred, t.cols = t.cols, type = 'occupancy')
str(out.pred)
```

sfJSDM   *Function for Fitting a Spatial Factor Joint Species Distribution Model*

Description

The function sfJSDM fits a spatially-explicit joint species distribution model. This model does not explicitly account for imperfect detection (see sfMsPGOcc()). We use Polya-Gamma latent variables and a spatial factor modeling approach. Currently, models are implemented using a Nearest Neighbor Gaussian Process.

Usage

```
sfJSDM(formula, data, inits, priors, tuning,
       cov.model = 'exponential', NNGP = TRUE,
       n.neighbors = 15, search.type = 'cb',
       std.by.sp = FALSE, n.factors, n.batch,
       batch.length, accept.rate = 0.43, n.omp.threads = 1,
       verbose = TRUE, n.report = 100,
       n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
       n.chains = 1, k.fold, k.fold.threads = 1, k.fold.seed = 100,
     k.fold.only = FALSE, monitors, keep.only.mean.95, shared.spatial = FALSE, ...)
```

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 are allowed using **lme4** syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, covs, coords, range.ind. y is a two-dimensional array with first dimension equal to the number of species and second dimension equal to the number of sites. Note how this differs from other spOccupancy functions in that y does not have any replicate surveys. This is because sfJSDM does not account for imperfect detection. covs is a matrix or data frame containing the variables used in the model, with J rows for each column (variable). coords is a matrix with J rows and 2 columns consisting of the spatial coordinates of each site in the data. Note that spOccupancy assumes coordinates are specified in a projected coordinate system. range.ind is a matrix with rows corresponding to species and columns corresponding to sites, with each element taking value 1 if that site is within the range of the corresponding species and 0 if it is outside of the range. This matrix is not required, but it can be helpful to restrict the modeled area for each individual species to be within the realistic range of locations for that species when estimating the model parameters.
inits	a list with each tag corresponding to a parameter name. Valid tags are beta.comm, beta, tau.sq.beta, phi, lambda, sigma.sq.psi, and nu. nu is only specified if cov.model = "matern". sigma.sq.psi is only specified if random intercepts 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, phi.unif, nu.unif, and sigma.sq.psi.ig. Community-

level occurrence (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 set to 2.73. 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 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.psi is the random effect variance for any random effects, and is 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.

tuning   a list with each tag corresponding to a parameter name. Valid tags are phi and nu. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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. 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 or k-fold cross-validation.

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.
std.by.sp	a logical value indicating whether the covariates are standardized separately for each species within the corresponding range for each species (TRUE) or not (FALSE). Note that if range.ind is specified in data.list, this will result in the covariates being standardized differently for each species based on the sites where range.ind == 1 for that given species. If range.ind is not specified and std.by.sp = TRUE, this will simply be equivalent to standardizing the covariates across all locations prior to fitting the model.
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 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.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015).

Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.

k.fold.threads   number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.

k.fold.seed   seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.

k.fold.only   a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.

monitors   a character vector used to indicate if only a subset of the model model parameters are desired to be monitored. If posterior samples of all parameters are desired, then don't specify the argument (this is the default). When working with a large number of species and/or sites, the full model object can be quite large, and so this argument can be used to only return samples of specific parameters to help reduce the size of this resulting object. Valid tags include beta.comm, tau.sq.beta, beta, z, psi, lambda, theta, w, like (used for WAIC calculation), beta.star, sigma.sq.psi. Note that if all parameters are not returned, subsequent functions that require the model object may not work. We only recommend specifying this option when working with large data sets (e.g., > 100 species and/or > 10,000 sites).

keep.only.mean.95
            not currently supported.

shared.spatial   a logical value used to specify whether a common spatial process should be estimated for all species instead of the factor modeling approach. If true, a spatial variance parameter sigma.sq is estimated for the model, which can be specified in the initial values and prior distributions (sigma.sq.ig).

...   currently no additional arguments

Value

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

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

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

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

theta.samples   a coda object of posterior samples for the species level correlation parameters.

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

psi.samples   a three-dimensional array of posterior samples for the latent occurrence probability values for each species.

w.samples   a three-dimensional array of posterior samples for the latent spatial random effects for each latent factor. Array dimensions correspond to MCMC sample, latent factor, and site. If shared.spatial = TRUE, this is still returned as a three-dimensional array where the first dimension is MCMC sample, second dimension is 1, and third dimension is site.

sigma.sq.psi.samples

     a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in formula.

beta.star.samples

     a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in 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().

k.fold.deviance

     vector of scoring rules (deviance) from k-fold cross-validation. A separate value is reported for each species. Only included if k.fold is specified in function call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

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

```
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep<- sample(2:4, size = J, replace = TRUE)
N <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6)
# Detection
alpha.mean <- c(0)
tau.sq.alpha <- c(1)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
alpha.true <- alpha
n.factors <- 3
phi <- rep(3 / .7, n.factors)
sigma.sq <- rep(2, n.factors)
nu <- rep(2, n.factors)

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                psi.RE = psi.RE, p.RE = p.RE, sp = TRUE, sigma.sq = sigma.sq,
                phi = phi, nu = nu, cov.model = 'matern', factor.model = TRUE,
```

```
                    n.factors = n.factors)

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]
coords <- as.matrix(dat$coords[-pred.indx, , drop = FALSE])
# Prediction covariates
X.0 <- dat$X[pred.indx, , drop = FALSE]
coords.0 <- as.matrix(dat$coords[pred.indx, , drop = FALSE])
# Detection covariates
X.p <- dat$X.p[-pred.indx, , , drop = FALSE]

y <- apply(y, c(1, 2), max, na.rm = TRUE)
data.list <- list(y = y, coords = coords)
# Priors
prior.list <- list(beta.comm.normal = list(mean = 0, var = 2.72),
                   tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                   nu.unif = list(0.5, 2.5))
# Starting values
inits.list <- list(beta.comm = 0,
                   beta = 0,
                   fix = TRUE,
                   tau.sq.beta = 1)
# Tuning
tuning.list <- list(phi = 1, nu = 0.25)

batch.length <- 25
n.batch <- 5
n.report <- 100
formula <- ~ 1

out <- sfJSDM(formula = formula,
              data = data.list,
              inits = inits.list,
              n.batch = n.batch,
              batch.length = batch.length,
              accept.rate = 0.43,
              priors = prior.list,
              cov.model = "matern",
              tuning = tuning.list,
              n.factors = 3,
              n.omp.threads = 1,
              verbose = TRUE,
              NNGP = TRUE,
              n.neighbors = 5,
              search.type = 'cb',
              n.report = 10,
              n.burn = 0,
              n.thin = 1,
              n.chains = 2)
summary(out)
```

---

sfMsPGOcc   *Function for Fitting Spatial Factor Multi-Species Occupancy Models*

Description

The function `sfMsPGOcc` fits multi-species spatial occupancy models with species correlations (i.e., a spatially-explicit joint species distribution model with imperfect detection). We use Polya-Gamma latent variables and a spatial factor modeling approach. Currently, models are implemented using a Nearest Neighbor Gaussian Process.

Usage

```
sfMsPGOcc(occ.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, n.omp.threads = 1,
          verbose = TRUE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
          n.chains = 1, k.fold, k.fold.threads = 1, k.fold.seed,
          k.fold.only = FALSE, ...)
```

Arguments

occ.formula	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Random intercepts are allowed using lme4 syntax (Bates et al. 2015). Only right-hand side of formula is specified. See example below.
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 are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, coords. y is a three-dimensional array 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. occ.covs is a matrix or data frame containing the variables used in the occurrence 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 spOccupancy 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.psi`, `sigma.sq.p`, `z`, `phi`, `lambda`, and `nu`. `nu` is only specified if `cov.model = "matern"`, and `sigma.sq.psi` and `sigma.sq.p` are only specified if random effects are included in `occ.formula` or `det.formula`, respectively. 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`, `tau.beta.half.t`, `tau.alpha.half.t`, `sigma.sq.psi`, `sigma.sq.p`, `phi.unif`, and `nu.unif`. Community-level occurrence (`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 set to 2.73. By default, community-level variance parameters for occupancy (`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 priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. Alternatively, half-t priors can be specified for the community level occurrence/detection standard deviation parameters using the tags `tau.beta.half.t` and `tau.alpha.half.t`. The hyperparameters of the half-t distribution are passed as a list of length two with the first and second elements corresponding to the degrees of freedom 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. 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 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.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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.

tuning   a list with each tag corresponding to a parameter name. Valid tags are phi and nu. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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. 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 or k-fold cross-validation.

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 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.

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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.comm.samples
> a coda object of posterior samples for the community level occurrence 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 occurrence 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 occurrence regression coefficients.

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

theta.samples   a coda object of posterior samples for the species level correlation parameters.

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

z.samples   a three-dimensional array of posterior samples for the latent occurrence values
                   for each species.

psi.samples   a three-dimensional array of posterior samples for the latent occupancy proba-
                   bility values for each species.

w.samples   a three-dimensional array of posterior samples for the latent spatial random ef-
                   fects for each latent factor.

sigma.sq.psi.samples
                   a coda object of posterior samples for variances of random intercepts included
                   in the occurrence portion of the model. Only included if random intercepts are
                   specified in occ.formula.

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

beta.star.samples
                   a coda object of posterior samples for the occurrence random effects. Only
                   included if random intercepts are specified in occ.formula.

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

like.samples   a three-dimensional array of posterior samples for the likelihood value associ-
                   ated 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().

k.fold.deviance
                   vector of scoring rules (deviance) from k-fold cross-validation. A separate value
                   is reported for each species. Only included if k.fold is specified in function
                   call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribu-
tion is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code
implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.
lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

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(400)

# Simulate Data -----------------------------------------------------------
J.x <- 7
J.y <- 7
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
N <- 8
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -.2)
tau.sq.alpha <- c(0.2, 0.3, 0.8)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
# Include a non-spatial random effect on occurrence
psi.RE <- list(levels = c(20),
               sigma.sq.psi = c(0.5))
p.RE <- list()
# Include a random effect on detection
p.RE <- list(levels = c(40),
    sigma.sq.p = c(2))
```

```
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
n.factors <- 4
phi <- runif(n.factors, 3/1, 3/.4)

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                phi = phi, sp = TRUE, cov.model = 'exponential',
                factor.model = TRUE, n.factors = n.factors, psi.RE = psi.RE,
                p.RE = p.RE)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.samples <- n.batch * batch.length

y <- dat$y
X <- dat$X
X.p <- dat$X.p
X.p.re <- dat$X.p.re
X.re <- dat$X.re
coords <- as.matrix(dat$coords)

# Package all data into a list
occ.covs <- cbind(X, X.re)
colnames(occ.covs) <- c('int', 'occ.cov', 'occ.re')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3],
                 det.re = X.p.re[, , 1])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
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, b = 3/.1))
# Initial values
lambda.inits <- matrix(0, N, n.factors)
diag(lambda.inits) <- 1
lambda.inits[lower.tri(lambda.inits)] <- rnorm(sum(lower.tri(lambda.inits)))

inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
```

```
                        beta = 0,
                        alpha = 0,
                        tau.sq.beta = 1,
                        tau.sq.alpha = 1,
                        phi = 3 / .5,
                        lambda = lambda.inits,
                        z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- sfMsPGOcc(occ.formula = ~ occ.cov + (1 | occ.re),
                 det.formula = ~ det.cov.1 + det.cov.2 + (1 | det.re),
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 accept.rate = 0.43,
                 priors = prior.list,
                 cov.model = "exponential",
                 tuning = tuning.list,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 NNGP = TRUE,
                 n.neighbors = 5,
                 n.factors = n.factors,
                 search.type = 'cb',
                 n.report = 10,
                 n.burn = 50,
                 n.thin = 1,
                 n.chains = 1)

summary(out)
```

---

simBinom   *Simulate Single-Species Binomial Data*

Description

The function `simBinom` simulates single-species binomial data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the model. Non-spatial random intercepts can also be included in the model.

Usage

```
simBinom(J.x, J.y, weights, beta, psi.RE = list(),
         sp = FALSE, svc.cols = 1, cov.model, sigma.sq, phi, nu,
         x.positive = FALSE, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate 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 data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
weights	a numeric vector of length $J = J.x \times J.y$ indicating the number of Bernoulli trials at each of the J sites.
beta	a numeric vector containing the intercept and regression coefficient parameters for the model.
psi.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.psi. 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.psi 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence 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. If svc.cols has more than one value, there should be a distinct spatial variance parameter for each spatially-varying coefficient.
phi	a numeric value indicating the spatial decay parameter. Ignored when sp = FALSE. If svc.cols has more than one value, there should be a distinct spatial decay parameter for each spatially-varying coefficient.
nu	a numeric value indicating the spatial smoothness parameter. Only used when sp = TRUE and cov.model = "matern". If svc.cols has more than one value, there should be a distinct spatial smoothness parameter for each spatially-varying coefficient.
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 $J \times p.occ$ numeric design matrix for the model.
coords	a $J \times 2$ numeric matrix of coordinates of each occupancy site. Required for spatial models.
w	a matrix of the spatial random effect values for each site. The number of columns is determined by the `svc.cols` argument (the number of spatially-varying coefficients).
psi	a $J \times 1$ matrix of the binomial probabilities for each site.
y	a length J vector of the binomial data for each site.
X.w	a two dimensional matrix containing the covariate effects (including an intercept) whose effects are assumed to be spatially-varying. Rows correspond to sites and columns correspond to covariate effects.
X.re	a numeric matrix containing the levels of any unstructured random effect included in the model. Only relevant when random effects are specified in `psi.RE`.
beta.star	a numeric vector that contains the simulated random effects for each given level of the random effects included in the model. Only relevant when random effects are included in the model.

Author(s)

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

Examples

```
set.seed(400)
J.x <- 10
J.y <- 10
weights <- rep(4, J.x * J.y)
beta <- c(0.5, -0.15)
svc.cols <- c(1, 2)
phi <- c(3 / .6, 3 / 0.2)
sigma.sq <- c(1.2, 0.9)
psi.RE <- list(levels = 10,
               sigma.sq.psi = 1.2)
dat <- simBinom(J.x = J.x, J.y = J.y, weights = weights, beta = beta,
                psi.RE = psi.RE, sp = TRUE, svc.cols = svc.cols,
                cov.model = 'spherical', sigma.sq = sigma.sq, phi = phi)
```

---

simIntMsOcc	*Simulate Multi-Species Detection-Nondetection Data from Multiple Data Sources*

Description

The function `simIntMsOcc` simulates multi-species detection-nondetection data from multiple data sources for simulation studies, power assessments, or function testing of integrated occupancy models. Data can optionally be simulated with a spatial Gaussian Process on the occurrence process.

Usage

```
simIntMsOcc(n.data, J.x, J.y, J.obs, n.rep, N, beta, alpha, psi.RE = list(),
            p.RE = list(), sp = FALSE, cov.model, sigma.sq, phi, nu,
            factor.model = FALSE, n.factors, ...)
```

Arguments

n.data	an integer indicating the number of detection-nondetection data sources to simulate.
J.x	a single numeric value indicating the number of sites across the region of interest along the horizontal axis. Total number of sites across the simulated region of interest is $J.x \times J.y$.
J.y	a single numeric value indicating the number of sites across the region of interest along the vertical axis. Total number of sites across the simulated region of interest is $J.x \times J.y$.
J.obs	a numeric vector of length n.data containing the number of sites to simulate each data source at. Data sources can be obtained at completely different sites, the same sites, or anywhere inbetween. Maximum number of sites a given data source is available at is equal to $J = J.x \times J.y$.
n.rep	a list of length n.data. Each element is a numeric vector with length corresponding to the number of sites that given data source is observed at (in J.obs). Each vector indicates the number of repeat visits at each of the sites for a given data source.
N	a numeric vector of length N containing the number of species each data source samples. These can be the same if both data sets sample the same species, or can be different.
beta	a numeric matrix with max(N) rows containing the intercept and regression coefficient parameters for the occurrence portion of the multi-species occupancy model. Each row corresponds to the regression coefficients for a given species.
alpha	a list of length n.data. Each element is a numeric matrix with the rows corresponding to the number of species that data source contains and columns corresponding to the regression coefficients for each data source.
psi.RE	a list used to specify the non-spatial random intercepts included in the occurrence portion of the model. The list must have two tags: levels and sigma.sq.psi. 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.psi 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 occurrence portion of the model.
p.RE	this argument is not currently supported. In a later version, this argument will allow for simulating data with detection random effects in the different data sources.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy 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 occurrence values. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
sigma.sq	a numeric vector of length max(N) containing the spatial variance parameter for each species. Ignored when sp = FALSE or when factor.model = TRUE.
phi	a numeric vector of length max(N) 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 max(N) 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.
...	currently no additional arguments

Value

A list comprised of:

X.obs	a numeric design matrix for the occurrence portion of the model. This matrix contains the intercept and regression coefficients for only the observed sites.
X.pred	a numeric design matrix for the occurrence portion of the model at sites where there are no observed data sources.
X.p	a list of design matrices for the detection portions of the integrated multi-species occupancy model. Each element in the list is a design matrix of detection covariates for each data source.
coords.obs	a numeric matrix of coordinates of each observed site. Required for spatial models.
coords.pred	a numeric matrix of coordinates of each site in the study region without any data sources. Only used for spatial models.
w	a species (or factor) x site 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.
w.pred	a matrix of the spatial random random effects for each species (or factor) at locations without any observation.
psi.obs	a species x site matrix of the occurrence probabilities for each species at the observed sites. Note that values are provided for all species, even if some species are only monitored at a subset of these points.
psi.pred	a species x site matrix of the occurrence probabilities for sites without any observations.

z.obs	a species x site matrix of the latent occurrence states at each observed site. Note that values are provided for all species, even if some species are only monitored at a subset of these points.
z.pred	a species x site matrix of the latent occurrence states at each site without any observations.
p	a list of detection probability arrays for each of the n.data data sources. Each array has dimensions corresponding to species, site, and replicate, respectively.
y	a list of arrays of the raw detection-nondetection data for each site and replicate combination for each species in the data set. Each array has dimensions corresponding to species, site, and replicate, respectively.

Author(s)

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

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

Examples

```
set.seed(91)
J.x <- 10
J.y <- 10
# Total number of data sources across the study region
J.all <- J.x * J.y
# Number of data sources.
n.data <- 2
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
n.rep <- list()
n.rep[[1]] <- rep(3, J.obs[1])
n.rep[[2]] <- rep(4, J.obs[2])

# Number of species observed in each data source
N <- c(8, 3)

# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, 0.5)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.4, 0.3)
# Detection
# Detection covariates
alpha.mean <- list()
tau.sq.alpha <- list()
# Number of detection parameters in each data source
```

```
p.det.long <- c(4, 3)
for (i in 1:n.data) {
  alpha.mean[[i]] <- runif(p.det.long[i], -1, 1)
  tau.sq.alpha[[i]] <- runif(p.det.long[i], 0.1, 1)
}
# Random effects
psi.RE <- list()
p.RE <- list()
beta <- matrix(NA, nrow = max(N), ncol = p.occ)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(max(N), beta.mean[i], sqrt(tau.sq.beta[i]))
}
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- matrix(NA, nrow = N[i], ncol = p.det.long[i])
  for (t in 1:p.det.long[i]) {
    alpha[[i]][, t] <- rnorm(N[i], alpha.mean[[i]][t], sqrt(tau.sq.alpha[[i]])[t])
  }
}
sp <- FALSE
factor.model <- FALSE
# Simulate occupancy data
dat <- simIntMsOcc(n.data = n.data, J.x = J.x, J.y = J.y,
   J.obs = J.obs, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
            psi.RE = psi.RE, p.RE = p.RE, sp = sp, factor.model = factor.model,
                  n.factors = n.factors)
str(dat)
```

---

simIntOcc	*Simulate Single-Species Detection-Nondetection Data from Multiple Data Sources*

Description

The function simIntOcc simulates single-species detection-nondetection data from multiple data sources for simulation studies, power assessments, or function testing of integrated occupancy models. Data can optionally be simulated with a spatial Gaussian Process on the occurrence process.

Usage

```
simIntOcc(n.data, J.x, J.y, J.obs, n.rep, n.rep.max, beta, alpha,
          sp = FALSE, cov.model, sigma.sq, phi, nu, ...)
```

Arguments

n.data	an integer indicating the number of detection-nondetection data sources to simulate.
J.x	a single numeric value indicating the number of sites across the region of interest along the horizontal axis. Total number of sites across the simulated region of interest is $J.x \times J.y$.

J.y   a single numeric value indicating the number of sites across the region of interest
                along the vertical axis. Total number of sites across the simulated region of
                interest is $J.x \times J.y$.

J.obs   a numeric vector of length n.data containing the number of sites to simulate
                each data source at. Data sources can be obtained at completely different sites,
                the same sites, or anywhere inbetween. Maximum number of sites a given data
                source is available at is equal to $J = J.x \times J.y$.

n.rep   a list of length n.data. Each element is a numeric vector with length corre-
                sponding to the number of sites that given data source is observed at (in J.obs).
                Each vector indicates the number of repeat visits at each of the sites for a given
                data source.

n.rep.max   a vector of numeric values indicating the maximum number of replicate surveys
                for each data set. This is an optional argument, with its default value set to
                max(n.rep) for each data set. This can be used to generate data sets with differ-
                ent 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 occurrence portion of the single-species occupancy model.

alpha   a list of length n.data. Each element is a numeric vector containing the in-
                tercept and regression coefficient parameters for the detection portion of the
                single-species occupancy model for each data source.

sp   a logical value indicating whether to simulate a spatially-explicit occupancy
                model with a Gaussian process. By default set to FALSE.

cov.model   a quoted keyword that specifies the covariance function used to model the spa-
                tial dependence structure among the latent occurrence values. Supported co-
                variance 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 range parameter. Ignored when sp =
                FALSE.

nu   a numeric value indicating the spatial smoothness parameter. Only used when
                sp = TRUE and cov.model = "matern".

...   currently no additional arguments

Value

A list comprised of:

X.obs   a numeric design matrix for the occurrence portion of the model. This matrix
                contains the intercept and regression coefficients for only the observed sites.

X.pred   a numeric design matrix for the occurrence portion of the model at sites where
                there are no observed data sources.

X.p   a list of design matrices for the detection portions of the integrated occupancy
                model. Each element in the list is a design matrix of detection covariates for
                each data source.

coords.obs	a numeric matrix of coordinates of each observed site. Required for spatial models.
coords.pred	a numeric matrix of coordinates of each site in the study region without any data sources. Only used for spatial models.
D.obs	a distance matrix of observed sites. Only used for spatial models.
D.pred	a distance matrix of sites in the study region without any observed data. Only used for spatial models.
w.obs	a matrix of the spatial random effects at observed locations. Only used to simulate data when sp = TRUE
.	
w.pred	a matrix of the spatial random random effects at locations without any observation.
psi.obs	a matrix of the occurrence probabilities for each observed site.
psi.pred	a matrix of the occurrence probabilities for sites without any observations.
z.obs	a vector of the latent occurrence states at each observed site.
z.pred	a vector of the latent occurrence states at each site without any observations.
p	a list of detection probability matrices for each of the n.data data sources.
y	a list of matrices of the raw detection-nondetection data for each site and replicate combination.

Author(s)

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

Examples

```
set.seed(400)

# Simulate Data ------------------------------------------------------------
J.x <- 15
J.y <- 15
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 1, -3)
p.occ <- length(beta)
# Detection covariates
```

```
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- runif(sample(1:4, 1), -1, 1)
}
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)
sigma.sq <- 2
phi <- 3 / .5
sp <- TRUE

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
                 n.rep = n.rep, beta = beta, alpha = alpha, sp = TRUE,
                 cov.model = 'gaussian', sigma.sq = sigma.sq, phi = phi)
```

---

simMsOcc   *Simulate Multi-Species Detection-Nondetection Data*

Description

The function simMsOcc simulates multi-species detection-nondetection data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the occurrence portion of the model, as well as an option to allow for species correlations using a factor modeling approach. Non-spatial random intercepts can also be included in the detection or occurrence portions of the occupancy model.

Usage

```
simMsOcc(J.x, J.y, n.rep, n.rep.max, N, beta, alpha, psi.RE = list(),
         p.RE = list(), sp = FALSE, svc.cols = 1, cov.model,
 sigma.sq, phi, nu, factor.model = FALSE, n.factors,
         range.probs, shared.spatial = FALSE, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate detection-nondetection 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 detection-nondetection 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	a single numeric value indicating the number of species to simulate detection-nondetection data.
beta	a numeric matrix with N rows containing the intercept and regression coefficient parameters for the occurrence portion of the multi-species occupancy model. Each row corresponds to the regression coefficients for a given species.
alpha	a numeric matrix with N rows containing the intercept and regression coefficient parameters for the detection portion of the multi-species occupancy model. Each row corresponds to the regression coefficients for a given species.
psi.RE	a list used to specify the non-spatial random intercepts included in the occurrence portion of the model. The list must have two tags: levels and sigma.sq.psi. 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.psi 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 occurrence 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence values. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
sigma.sq	a numeric vector of length N containing the spatial variance parameter for each species. Ignored when sp = FALSE or when factor.model = TRUE.
phi	a numeric vector of length N 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 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.
range.probs	a numeric vector of length N where each value should fall between 0 and 1, and indicates the probability that one of the J spatial locations simulated is within the simulated range of the given species. If set to 1, every species has the potential of being present at each location.
shared.spatial	a logical value indicating used to specify whether a common spatial process should be estimated for all species instead of the factor modeling approach.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times p.occ$ numeric design matrix for the occurrence 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 occupancy model.
coords	a $J \times 2$ numeric matrix of coordinates of each occupancy 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.
psi	a $N \times J$ matrix of the occurrence probabilities for each species at each site.
z	a $N \times J$ matrix of the latent occurrence states for each species at each site.
p	a N 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 x J x max(n.rep) array of the raw detection-nondetection 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.lambda.re	a numeric matrix containing the levels of any occurrence random effect included in the model. Only relevant when occurrence random effects are specified in psi.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 occurrence random effects for each given level of the random effects included in the occurrence model. Only relevant when occurrence random effects are included in the model.

Author(s)

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

Examples

```
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
N <- 10
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- 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)
psi.RE <- list(levels = c(10),
               sigma.sq.psi = c(1.5))
p.RE <- list(levels = c(15),
             sigma.sq.p = 0.8)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
# Spatial parameters if desired
phi <- runif(N, 3/1, 3/.1)
sigma.sq <- runif(N, 0.3, 3)
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta,
                alpha = alpha, psi.RE = psi.RE, p.RE = p.RE, sp = TRUE,
                cov.model = 'exponential', phi = phi, sigma.sq = sigma.sq)
```

---

simOcc   *Simulate Single-Species Detection-Nondetection Data*

Description

The function simOcc simulates single-species occurrence data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the occurrence portion of the model. Non-spatial random intercepts can also be included in the detection or occurrence portions of the occupancy model.

Usage

```
simOcc(J.x, J.y, n.rep, n.rep.max, beta, alpha, psi.RE = list(),
       p.RE = list(), sp = FALSE, svc.cols = 1, cov.model,
       sigma.sq, phi, nu, x.positive = FALSE, grid, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate detection-nondetection 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 detection-nondetection 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 occupancy portion of the single-species occupancy model.
alpha	a numeric vector containing the intercept and regression coefficient parameters for the detection portion of the single-species occupancy model.
psi.RE	a list used to specify the non-spatial random intercepts included in the occupancy portion of the model. The list must have two tags: levels and sigma.sq.psi. 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.psi 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 occupancy 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence 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".
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).
grid	a list or atomic vector used to specify the grid across which to simulate the latent spatial processes. If a list, each element should correspond to the grid used for each SVC in the model. This argument is used to simulate the underlying spatial process at a different resolution than the coordinates (e.g., if coordinates are distributed across a grid).
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times p.occ$ numeric design matrix for the occupancy 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 occupancy model.
coords	a $J \times 2$ numeric matrix of coordinates of each occupancy site. Required for spatial models.
w	a matrix of the spatial random effect values for each site. The number of columns is determined by the svc.cols argument (the number of spatially-varying coefficients).
psi	a $J \times 1$ matrix of the occupancy probabilities for each site.
z	a length J vector of the latent occupancy states 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 detection-nondetection 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 occurrence random effect included in the model. Only relevant when occurrence random effects are specified in psi.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 occurrence random effects for each given level of the random effects included in the occurrence model. Only relevant when occurrence random effects are included in the model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@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)
phi <- 3 / .6
sigma.sq <- 2
psi.RE <- list(levels = 10,
               sigma.sq.psi = 1.2)
p.RE <- list(levels = 15,
             sigma.sq.p = 0.8)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              psi.RE = psi.RE, p.RE = p.RE, sp = TRUE, cov.model = 'spherical',
              sigma.sq = sigma.sq, phi = phi)
```

---

simTBinom   *Simulate Multi-Season Single-Species Binomial Data*

Description

The function simTBinom simulates multi-season single-species binomial data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the model. Non-spatial random intercepts can also be included in the model.

Usage

```
simTBinom(J.x, J.y, n.time, weights, beta, sp.only = 0,
          trend = TRUE, psi.RE = list(), sp = FALSE,
          cov.model, sigma.sq, phi, nu, svc.cols = 1,
          ar1 = FALSE, rho, sigma.sq.t, x.positive = FALSE, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate 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 data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
n.time	a single numeric value indicating the number of primary time periods (denoted T) over which sampling occurs.
weights	a numeric matrix with rows corresponding to sites and columns corresponding to primary time periods that indicates the number of Bernoulli trials at each of the site/time period combinations.
beta	a numeric vector containing the intercept and regression coefficient parameters for the model.
sp.only	a numeric vector specifying which occurrence covariates should only vary over space and not over time. The numbers in the vector correspond to the elements in the vector of regression coefficients (beta). By default, all simulated occurrence covariates are assumed to vary over both space and time.
trend	a logical value. If TRUE, a temporal trend will be used to simulate the detection-nondetection data and the second element of beta is assumed to be the trend parameter. If FALSE no trend is used to simulate the data and all elements of beta (except the first value which is the intercept) correspond to covariate effects.
psi.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.psi. 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.psi 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence 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. If svc.cols has more than one value, there should be a distinct spatial variance parameter for each spatially-varying coefficient.
phi	a numeric value indicating the spatial decay parameter. Ignored when sp = FALSE. If svc.cols has more than one value, there should be a distinct spatial decay parameter for each spatially-varying coefficient.

nu	a numeric value indicating the spatial smoothness parameter. Only used when sp = TRUE and cov.model = "matern". If svc.cols has more than one value, there should be a distinct spatial smoothness parameter for each spatially-varying co-efficient.
ar1	a logical value indicating whether to simulate a temporal random effect with an AR(1) process. By default, set to FALSE.
rho	a numeric value indicating the AR(1) temporal correlation parameter. Ignored when ar1 = FALSE.
sigma.sq.t	a numeric value indicating the AR(1) temporal variance parameter. Ignored when ar1 = FALSE.
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 (x.positive = TRUE). If x.positive = TRUE, covariates are simulated from a random normal and then the minimum value is added to each covariate value to ensure non-negative covariate values.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times T \times p.occ$ numeric array containing the design matrix for the model.
coords	a $J \times 2$ numeric matrix of coordinates of each occupancy site. Required for spatial models.
w	a matrix of the spatial random effect values for each site. The number of columns is determined by the svc.cols argument (the number of spatially-varying coefficients).
psi	a $J \times T$ matrix of the occupancy probabilities for each site during each primary time period.
z	a $J \times T$ matrix of the binomial data at each site during each primary time period.
X.w	a three dimensional array containing the covariate effects (including an intercept) whose effects are assumed to be spatially-varying. Dimensions correspond to sites, primary time periods, and covariate.
X.re	a numeric matrix containing the levels of any unstructured random effect included in the model. Only relevant when random effects are specified in psi.RE.
beta.star	a numeric vector that contains the simulated random effects for each given level of the random effects included in the model. Only relevant when random effects are included in the model.

Author(s)

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

Examples

```
set.seed(1000)
# Sites
J.x <- 15
J.y <- 15
J <- J.x * J.y
# Years sampled
n.time <- sample(10, J, replace = TRUE)
# Binomial weights
weights <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  weights[j, 1:n.time[j]] <- sample(5, n.time[j], replace = TRUE)
}
# Occurrence -------------------------
beta <- c(-2, -0.5, -0.2, 0.75)
p.occ <- length(beta)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Spatial parameters -----------------
sp <- TRUE
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.1, 1)
phi <- runif(p.svc, 3/1, 3/0.2)
# Temporal parameters ----------------
ar1 <- TRUE
rho <- 0.8
sigma.sq.t <- 1

dat <- simTBinom(J.x = J.x, J.y = J.y, n.time = n.time, weights = weights, beta = beta,
                 psi.RE = psi.RE, sp.only = sp.only, trend = trend,
                 sp = sp, svc.cols = svc.cols,
                 cov.model = cov.model, sigma.sq = sigma.sq, phi = phi,
                 rho = rho, sigma.sq.t = sigma.sq.t, ar1 = TRUE, x.positive = FALSE)
```

---

simTMsOcc   *Simulate Multi-Species Multi-Season Detection-Nondetection Data*

Description

The function `simTMsOcc` simulates multi-species multi-season detection-nondetection data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the occurrence portion of the model, as well as an option to allow for species correlations using a factor modeling approach. Non-spatial random intercepts can also be included in the detection or occurrence portions of the occupancy model.

Usage

```
simTMsOcc(J.x, J.y, n.time, n.rep, N, beta, alpha, sp.only = 0,
  trend = TRUE, psi.RE = list(), p.RE = list(),
          sp = FALSE, svc.cols = 1, cov.model,
  sigma.sq, phi, nu, ar1 = FALSE, rho, sigma.sq.t,
  factor.model = FALSE, n.factors, range.probs, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate detection-nondetection 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 detection-nondetection data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
n.time	a single numeric value indicating the number of primary time periods (denoted T) over which sampling occurs.
n.rep	a numeric matrix indicating the number of replicates at each site during each primary time period. The matrix must have $J = J.x \times J.y$ rows and T columns, where T is the number of primary time periods (e.g., years or seasons) over which sampling occurs.
N	a single numeric value indicating the number of species to simulate detection-nondetection data.
beta	a numeric matrix with N rows containing the intercept and regression coefficient parameters for the occurrence portion of the multi-species occupancy model. Each row corresponds to the regression coefficients for a given species.
alpha	a numeric matrix with N rows containing the intercept and regression coefficient parameters for the detection portion of the multi-species occupancy model. Each row corresponds to the regression coefficients for a given species.
sp.only	a numeric vector specifying which occurrence covariates should only vary over space and not over time. The numbers in the vector correspond to the elements in the vector of regression coefficients (beta). By default, all simulated occurrence covariates are assumed to vary over both space and time.
trend	a logical value. If TRUE, a temporal trend will be used to simulate the detection-nondetection data and the second element of beta is assumed to be the trend parameter. If FALSE no trend is used to simulate the data and all elements of beta (except the first value which is the intercept) correspond to covariate effects.
psi.RE	a list used to specify the non-spatial random intercepts included in the occurrence portion of the model. The list must have two tags: levels and sigma.sq.psi. 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.psi 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 occurrence 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence values. Supported covariance model key words are: "exponential", "matern", "spherical", and "gaussian".
sigma.sq	a numeric vector of length N containing the spatial variance parameter for each species. Ignored when sp = FALSE or when factor.model = TRUE.
phi	a numeric vector of length N 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 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.
ar1	a logical value indicating whether to simulate a temporal random effect with an AR(1) process. By default, set to FALSE.
rho	a vector of N values indicating the AR(1) temporal correlation parameter for each species. Ignored when ar1 = FALSE.
sigma.sq.t	a vector of N values indicating the AR(1) temporal variance parameter for each species. Ignored when ar1 = FALSE.
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.
range.probs	a numeric vector of length N where each value should fall between 0 and 1, and indicates the probability that one of the J spatial locations simulated is within the simulated range of the given species. If set to 1, every species has the potential of being present at each location.
...	currently no additional arguments

Value

A list comprised of:

X	a $J \times T \times p.occ$ numeric array containing the design matrix for the occurrence portion of the occupancy model.
X.p	a four-dimensional numeric array with dimensions corresponding to sites, primary time periods, repeat visits, and number of detection regression coefficients. This is the design matrix used for the detection portion of the occupancy model.
coords	a $J \times 2$ numeric matrix of coordinates of each occupancy 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.
psi	a $N \times J \times T$ array of the occurrence probabilities for each species at each site during each primary time period.
z	a $N \times J \times T$ array of the latent occurrence status for each species at each site during each primary time period.
p	a N x J x T x max(n.rep) array of the detection probabilities for each species at each site, primary time period, and secondyary replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
y	a N x J x T x max(n.rep) array of the raw detection-nondetection data for each species at each site, primary time period, and replicate combination. Sites with fewer than max(n.rep) replicates will contain NA values.
X.p.re	a four-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 occurrence random effect included in the model. Only relevant when occurrence random effects are specified in psi.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 occurrence random effects for each given level of the random effects included in the occurrence model. Only relevant when occurrence random effects are included in the model.
eta	a numeric matrix with each row corresponding to species and column corresponding to time period of the AR(1) temporal random effects.

Author(s)

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

Examples

```
# Simulate Data -----------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
  # n.rep[j, 1:n.time[j]] <- rep(4, n.time[j])
}
N <- 7
# Community-level covariate effects
# Occurrence
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
n.factors <- 3
phi <- runif(p.svc * n.factors, 3 / .9, 3 / .3)
factor.model <- TRUE
cov.model <- 'exponential'

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
 psi.RE = psi.RE, p.RE = p.RE, factor.model = factor.model,
                 svc.cols = svc.cols, n.factors = n.factors, phi = phi, sp = sp,
```

```
                          cov.model = cov.model)
    str(dat)
```

---

simTOcc   *Simulate Multi-Season Single-Species Detection-Nondetection Data*

Description

The function simTOcc simulates multi-season single-species occurrence data for simulation studies, power assessments, or function testing. Data can be optionally simulated with a spatial Gaussian Process in the occurrence portion of the model. Non-spatial random intercepts can also be included in the detection or occurrence portions of the occupancy model.

Usage

```
simTOcc(J.x, J.y, n.time, n.rep, n.rep.max, beta, alpha, sp.only = 0, trend = TRUE,
        psi.RE = list(), p.RE = list(), sp = FALSE, svc.cols = 1, cov.model,
        sigma.sq, phi, nu, ar1 = FALSE, rho, sigma.sq.t, x.positive = FALSE,
mis.spec.type = 'none', scale.param = 1, avail, ...)
```

Arguments

J.x	a single numeric value indicating the number of sites to simulate detection-nondetection 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 detection-nondetection data along the vertical axis. Total number of sites with simulated data is $J.x \times J.y$.
n.time	a single numeric value indicating the number of primary time periods (denoted T) over which sampling occurs.
n.rep	a numeric matrix indicating the number of replicates at each site during each primary time period. The matrix must have $J = J.x \times J.y$ rows and T columns, where T is the number of primary time periods (e.g., years or seasons) over which sampling occurs.
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 occupancy portion of the single-species occupancy model. Note that if trend = TRUE, the second value in the vector corresponds to the estimated occurrence trend.
alpha	a numeric vector containing the intercept and regression coefficient parameters for the detection portion of the single-species occupancy model.

sp.only	a numeric vector specifying which occurrence covariates should only vary over space and not over time. The numbers in the vector correspond to the elements in the vector of regression coefficients (beta). By default, all simulated occurrence covariates are assumed to vary over both space and time.
trend	a logical value. If TRUE, a temporal trend will be used to simulate the detection-nondetection data and the second element of beta is assumed to be the trend parameter. If FALSE no trend is used to simulate the data and all elements of beta (except the first value which is the intercept) correspond to covariate effects.
psi.RE	a list used to specify the unstructured random intercepts included in the occupancy portion of the model. The list must have two tags: levels and sigma.sq.psi. 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.psi 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. An additional tag site.RE can be set to TRUE to simulate data with a site-specific non-spatial random effect on occurrence. If not specified, no random effects are included in the occupancy portion of the model.
p.RE	a list used to specify the unstructured 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.
sp	a logical value indicating whether to simulate a spatially-explicit occupancy model with a Gaussian process. By default set to FALSE.
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 latent occurrence 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".
ar1	a logical value indicating whether to simulate a temporal random effect with an AR(1) process. By default, set to FALSE.
rho	a numeric value indicating the AR(1) temporal correlation parameter. Ignored when ar1 = FALSE.

sigma.sq.t   a numeric value indicating the AR(1) temporal variance parameter. Ignored
                   when ar1 = FALSE.

x.positive   a logical value indicating whether the simulated covariates should be simu-
                   lated as random standard normal covariates (x.positive = FALSE) or restricted
                   to positive values (x.positive = TRUE). If x.positive = TRUE, covariates are
                   simulated from a random normal and then the minimum value is added to each
                   covariate value to ensure non-negative covariate values.

mis.spec.type   a quoted keyword indicating the type of model mis-specification to use when
                   simulating the data. These correspond to model mis-specification of the func-
                   tional relationship between occupancy/detection probability and covariates. Valid
                   keywords are: "none" (no model mis-specification, i.e., logit link), "scale"
                   (scaled logistic link), "line" (linear link), and "probit" (probit link). Defaults
                   to "none".

scale.param   a positive number between 0 and 1 that indicates the scale parameter for the
                   occupancy portion of the model when mis.spec.type = 'scale'. When speci-
                   fied, scale.param corresponds to the scale parameter for the occupancy portion
                   of the model, while the reciprocal of scale.param is used for the detection por-
                   tion of the model.

avail   a site x primary time period x visit array indicating the availability probabil-
                   ity of the species during each survey simulated at the given site/primary time
                   period/visit combination. This can be used to assess impacts of non-constant
                   availability across replicate surveys in simulation studies. Values should fall
                   between 0 and 1. When not specified, availability is set to 1 for all surveys.

...   currently no additional arguments

Value

A list comprised of:

X   a $J \times T \times p.occ$ numeric array containing the design matrix for the occurrence
                   portion of the occupancy model.

X.p   a four-dimensional numeric array with dimensions corresponding to sites, pri-
                   mary time periods, repeat visits, and number of detection regression coefficients.
                   This is the design matrix used for the detection portion of the occupancy model.

coords   a $J \times 2$ numeric matrix of coordinates of each occupancy site. Required for
                   spatial models.

w   a $J \times 1$ matrix of the spatial random effects. Only used to simulate data when
                   sp = TRUE.

psi   a $J \times T$ matrix of the occupancy probabilities for each site during each primary
                   time period.

z   a $J \times T$ matrix of the latent occupancy states at each site during each primary
                   time period.

p   a J x T x max(n.rep) array of the detection probabilities for each site, primary
                   time period, and replicate combination. Site/time periods with fewer than max(n.rep)
                   replicates will contain NA values.

y	a J x T x max(n.rep) array of the raw detection-nondetection data for each sit, primary time period, and replicate combination.
X.p.re	a four-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 occurrence random effect included in the model. Only relevant when occurrence random effects are specified in psi.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 occurrence random effects for each given level of the random effects included in the occurrence model. Only relevant when occurrence random effects are included in the model.
eta	a $T \times 1$ matrix of the latent AR(1) random effects. Only included when ar1 = TRUE.

Author(s)

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

References

Stoudt, S., P. de Valpine, and W. Fithian. Non-parametric identifiability in species distribution and abundance models: why it matters and how to diagnose a lack of fit using simulation. Journal of Statistical Theory and Practice 17, 39 (2023). https://doi.org/10.1007/s42519-023-00336-5.

Examples

```
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Number of time periods sampled
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence ------------------------
# Fixed
beta <- c(0.4, 0.5, -0.9)
trend <- TRUE
sp.only <- 0
psi.RE <- list(levels = c(10),
               sigma.sq.psi = c(1))
# Detection -------------------------
```

```
alpha <- c(-1, 0.7, -0.5)
p.RE <- list(levels = c(10),
             sigma.sq.p = c(0.5))
# Spatial parameters -----------------
sp <- TRUE
cov.model <- "exponential"
sigma.sq <- 2
phi <- 3 / .4
nu <- 1
# Temporal parameters ----------------
ar1 <- TRUE
rho <- 0.5
sigma.sq.t <- 0.8
# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
               beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
               psi.RE = psi.RE, p.RE = p.RE,
               sp = sp, cov.model = cov.model, sigma.sq = sigma.sq, phi = phi,
               ar1 = ar1, rho = rho, sigma.sq.t = sigma.sq.t)
str(dat)
```

---

spIntPGOcc   *Function for Fitting Single-Species Integrated Spatial Occupancy*
                            Models Using Polya-Gamma Latent Variables

---

Description

The function `spIntPGOcc` fits single-species integrated spatial occupancy models using Polya-
Gamma latent variables. Models can be fit using either a full Gaussian process or a Nearest Neigh-
bor Gaussian Process for large data sets. Data integration is done using a joint likelihood framework,
assuming distinct detection models for each data source that are each conditional on a single latent
occupancy process.

Usage

```
spIntPGOcc(occ.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, n.omp.threads = 1,
           verbose = TRUE, n.report = 100,
           n.burn = round(.10 * n.batch * batch.length),
           n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
           k.fold.seed, k.fold.data, k.fold.only = FALSE, ...)
```

Arguments

occ.formula   a symbolic description of the model to be fit for the occurrence portion of the
                 model using R's model syntax. Only right-hand side of formula is specified. See
                 example below.

det.formula   a list of symbolic descriptions of the models to be fit for the detection portion of the model using R's model syntax for each data set. Each element in the list is a formula for the detection model of a given data set. Only right-hand side of formula is specified. See example below.

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, sites and coords. y is a list of matrices or data frames for each data set used in the integrated model. Each element of the list has first dimension equal to the number of sites with that data source and second dimension equal to the maximum number of replicates at a given site. occ.covs is a matrix or data frame containing the variables used in the occurrence portion of the model, with the number of rows being the number of sites with at least one data source for each column (variable). det.covs is a list of variables included in the detection portion of the model for each data source. det.covs should have the same number of elements as y, where each element is itself a list. Each element of the list for a given data source is a different detection covariate, which can be site-level or observational-level. Site-level covariates are specified as a vector with length equal to the number of observed sites of that data source, while observation-level covariates are specified as a matrix or data frame with the number of rows equal to the number of observed sites of that data source and number of columns equal to the maximum number of replicates at a given site. coords is a matrix of the observation site coordinates. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta, alpha, sigma.sq, phi, w, and nu. The value portion of all tags except alpha is the parameter's initial value. The tag alpha is a list comprised of the initial values for the detection parameters for each data source. Each element of the list should be a vector of initial values for all detection parameters in the given data source or a single value for each data source to assign all parameters for a given data source the same 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, sigma.sq.unif, and nu.unif. Occurrence (beta) and detection (alpha) regression coefficients are assumed to follow a normal distribution. For beta 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. For the detection coefficients alpha, the mean and variance hyperparameters are themselves passed in as lists, with each element of the list corresponding to the specific hyperparameters for the detection parameters in a given data source. If not specified, prior means are set to 0 and prior variances set to 2.73 for normal priors. The spatial variance parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution or a uniform distribution (default is inverse-Gamma). sigma.sq can also be fixed at its initial value by setting the prior value to "fixed". The spatial decay phi and smoothness nu parameters are assumed to follow Uniform dis-

	tributions. The hyperparameters of the inverse-Gamma 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.
tuning	a list with each tag corresponding to a parameter name. Valid tags are phi and nu. The value portion of each tag defines the initial variance of the Adaptive sampler. 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. If FALSE, a full Gaussian process is used. See Datta et al. (2016) and Finley et al. (2019) for more information.
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 or k-fold cross-validation.
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 to run for each chain 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.
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.batch * batch.length samples 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 chains to run in sequence.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.data	an integer specifying the specific data set to hold out values from. If not specified, data from all data set locations will be incorporated into the k-fold cross-validation.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the occurrence regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients for all data sources.
z.samples	a coda object of posterior samples for the latent occurrence values
psi.samples	a coda object of posterior samples for the latent occurrence probability values
theta.samples	a coda object of posterior samples for covariance parameters.
w.samples	a coda object of posterior samples for latent spatial random effects.
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().
k.fold.deviance	
	scoring rule (deviance) from k-fold cross-validation. A separate deviance value is returned for each data source. Only included if k.fold is specified in function call. Only a single value is returned if k.fold.data is specified.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories. lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

Hooten, M. B., and Hefley, T. J. (2019). Bringing Bayesian models to life. *CRC Press*.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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(400)

# Simulate Data -------------------------------------------------------
# Number of locations in each direction. This is the total region of interest
# where some sites may or may not have a data source.
J.x <- 8
J.y <- 8
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
```

```
# Occupancy covariates
beta <- c(0.5, 0.5)
p.occ <- length(beta)
# Detection covariates
alpha <- list()
alpha[[1]] <- runif(2, 0, 1)
alpha[[2]] <- runif(3, 0, 1)
alpha[[3]] <- runif(2, -1, 1)
alpha[[4]] <- runif(4, -1, 1)
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)
sigma.sq <- 2
phi <- 3 / .5
sp <- TRUE

# Simulate occupancy data from multiple data sources.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
                 n.rep = n.rep, beta = beta, alpha = alpha, sp = sp,
                 sigma.sq = sigma.sq, phi = phi, cov.model = 'exponential')

y <- dat$y
X <- dat$X.obs
X.p <- dat$X.p
sites <- dat$sites
X.0 <- dat$X.pred
psi.0 <- dat$psi.pred
coords <- as.matrix(dat$coords.obs)
coords.0 <- as.matrix(dat$coords.pred)

# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list()
# Add covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2],
                      det.cov.2.2 = X.p[[2]][, , 3])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2],
                      det.cov.4.2 = X.p[[4]][, , 3],
                      det.cov.4.3 = X.p[[4]][, , 4])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites,
                  coords = coords)

J <- length(dat$z.obs)

# Initial values
inits.list <- list(alpha = list(0, 0, 0, 0),
                   beta = 0,
                   phi = 3 / .5,
```

```
                           sigma.sq = 2,
                           w = rep(0, J),
                           z = rep(1, J))
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = list(0, 0, 0, 0),
                                       var = list(2.72, 2.72, 2.72, 2.72)),
                   phi.unif = c(3/1, 3/.1),
                   sigma.sq.ig = c(2, 2))
# Tuning
tuning.list <- list(phi = 0.3)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25

out <- spIntPGOcc(occ.formula = ~ occ.cov,
                  det.formula = list(f.1 = ~ det.cov.1.1,
                                     f.2 = ~ det.cov.2.1 + det.cov.2.2,
                                     f.3 = ~ det.cov.3.1,
                                     f.4 = ~ det.cov.4.1 + det.cov.4.2 + det.cov.4.3),
                  data = data.list,
                  inits = inits.list,
                  n.batch = n.batch,
                  batch.length = batch.length,
                  accept.rate = 0.43,
                  priors = prior.list,
                  cov.model = "exponential",
                  tuning = tuning.list,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  NNGP = FALSE,
                  n.report = 10,
                  n.burn = 50,
                  n.thin = 1)

summary(out)
```

---

spMsPGOcc   *Function for Fitting Multi-Species Spatial Occupancy Models Using*
                         Polya-Gamma Latent Variables

Description

The function spMsPGOcc fits multi-species spatial occupancy models using Polya-Gamma latent
variables. Models can be fit using either a full Gaussian process or a Nearest Neighbor Gaussian
Process for large data sets.

Usage

```
spMsPGOcc(occ.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, n.omp.threads = 1,
          verbose = TRUE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
          n.chains = 1, k.fold, k.fold.threads = 1, k.fold.seed,
          k.fold.only = FALSE, ...)
```

Arguments

occ.formula	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Only right-hand side of formula is specified. See example below. Random intercepts 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 are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, coords. y is a three-dimensional array 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. occ.covs is a matrix or data frame containing the variables used in the occurrence 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 spOccupancy 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.psi, sigma.sq.p, z, sigma.sq, phi, w, and nu. nu is only specified if cov.model = "matern", sigma.sq.psi is only specified if there are random intercepts in occ.formula, and sigma.sq.p is only specified if there are random intercpets in det.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, alpha.comm.normal, tau.sq.beta.ig, tau.sq.alpha.ig, phi.unif, sigma.sq.ig, sigma.sq.unif, nu.unif, sigma.sq.psi, sigma.sq.p. Community-level occurrence (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 set to 2.73. Community-level variance parameters for occupancy (`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 priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. The species-specific spatial variance parameter, `sigma.sq`, is assumed to follow an inverse-Gamma distribution or a uniform distribution (default is inverse-Gamma). `sigma.sq` of all species can also be fixed at its initial value by setting the prior value to `"fixed"`. The spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma are passed as a list of length two, with the list elements being vectors of length N corresponding to the species-specific shape and scale parameters, respectively, or a single value if the same value is assigned for all species. The hyperparameters of the Uniform are also passed as a list with two elements, with both elements being vectors of length N corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all species. `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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.

tuning
a list with each tag corresponding to a parameter name. Valid tags are `phi` and `nu`. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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 keywords 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.

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 or k-fold cross-validation.

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 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.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

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

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

`theta.samples`   a coda object of posterior samples for the species level covariance parameters.

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

`psi.samples`   a three-dimensional array of posterior samples for the latent occupancy probability values for each species.

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

`sigma.sq.psi.samples`
> a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in `occ.formula`.

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

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

`beta.star.samples`
> a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in `occ.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().

k.fold.deviance

            vector of scoring rules (deviance) from k-fold cross-validation. A separate value is reported for each species. Only included if k.fold is specified in function call.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

Examples

```
set.seed(400)

# Simulate Data -------------------------------------------------------
J.x <- 7
J.y <- 7
```

```
J <- J.x * J.y
n.rep <- sample(2:4, size = J, replace = TRUE)
N <- 5
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.15)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 0.3)
# Detection
alpha.mean <- c(0.5, 0.2, -.2)
tau.sq.alpha <- c(0.2, 0.3, 0.8)
p.det <- length(alpha.mean)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
phi <- runif(N, 3/1, 3/.4)
sigma.sq <- runif(N, 0.3, 3)
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                phi = phi, sigma.sq = sigma.sq, sp = TRUE, cov.model = 'exponential')

# Number of batches
n.batch <- 30
# Batch length
batch.length <- 25
n.samples <- n.batch * batch.length

y <- dat$y
X <- dat$X
X.p <- dat$X.p
coords <- as.matrix(dat$coords)

# Package all data into a list
occ.covs <- X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
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, b = 3/.1),
                      sigma.sq.ig = list(a = 2, b = 2))
# Initial values
inits.list <- list(alpha.comm = 0,
                    beta.comm = 0,
                    beta = 0,
                    alpha = 0,
                    tau.sq.beta = 1,
                    tau.sq.alpha = 1,
                    phi = 3 / .5,
                    sigma.sq = 2,
                    w = matrix(0, nrow = N, ncol = nrow(X)),
                    z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- spMsPGOcc(occ.formula = ~ occ.cov,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 accept.rate = 0.43,
                 priors = prior.list,
                 cov.model = "exponential",
                 tuning = tuning.list,
                 n.omp.threads = 1,
                 verbose = TRUE,
                 NNGP = TRUE,
                 n.neighbors = 5,
                 search.type = 'cb',
                 n.report = 10,
                 n.burn = 500,
                 n.thin = 1,
                 n.chains = 1)

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

---

spPGOcc   *Function for Fitting Single-Species Spatial Occupancy Models Using Polya-Gamma Latent Variables*

Description

The function spPGOcc fits single-species spatial occupancy models using Polya-Gamma latent variables. Models can be fit using either a full Gaussian process or a Nearest Neighbor Gaussian Process for large data sets.

Usage

```
spPGOcc(occ.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,
        n.omp.threads = 1, verbose = TRUE, n.report = 100,
        n.burn = round(.10 * n.batch * batch.length),
        n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
        k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
               det.covs, and coords. y is the detection-nondetection data matrix or data
               frame with first dimension equal to the number of sites (J) and second dimen-
               sion equal to the maximum number of replicates at a given site. occ.covs is
               a matrix or data frame containing the variables used in the occupancy 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 spOccupancy assumes coordinates are specified in a projected coordinate
               system.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta,
               alpha, sigma.sq, phi, w, nu, sigma.sq.psi, sigma.sq.p. nu is only specified
               if cov.model = "matern", sigma.sq.p is only specified if there are random ef-
               fects in det.formula, and sigma.sq.psi is only specified if there are random
               effects in occ.formula. The value portion of each tag is the parameter's initial
               value. See priors description for definition of each parameter name. Addition-
               ally, 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, sigma.sq.unif, nu.unif, sigma.sq.psi.ig,
               and sigma.sq.p.ig. Occurrence (beta) and detection (alpha) regression co-
               efficients 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 2.73. The spatial variance parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution or a uniform distribution (default is inverse-Gamma). sigma.sq can also be fixed at its initial value by setting the prior value to "fixed". The spatial decay phi and smoothness nu 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.psi and sigma.sq.p are the random effect variances for any occurrence 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 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 list with each tag corresponding to a parameter name. Valid tags are phi and nu. The value portion of each tag defines the initial variance of the Adaptive sampler. See Roberts and Rosenthal (2009) for details.

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.

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 or k-fold cross-validation.

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.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the occurrence regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
z.samples	a coda object of posterior samples for the latent occurrence values
psi.samples	a coda object of posterior samples for the latent occurrence probability values
theta.samples	a coda object of posterior samples for covariance parameters.

w.samples   a coda object of posterior samples for latent spatial random effects.

sigma.sq.psi.samples

a coda object of posterior samples for variances of random intercepts included in the occupancy portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in det.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().

k.fold.deviance

soring rule (deviance) from k-fold cross-validation. Only included if k.fold is specified in function call.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

Hooten, M. B., and Hefley, T. J. (2019). Bringing Bayesian models to life. *CRC Press*.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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(350)
# Simulate Data -----------------------------------------------------------
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, -0.15)
p.occ <- length(beta)
alpha <- c(0.7, 0.4, -0.2)
p.det <- length(alpha)
phi <- 3 / .6
sigma.sq <- 2
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sigma.sq = sigma.sq, phi = phi, sp = TRUE, cov.model = 'exponential')
y <- dat$y
X <- dat$X
X.p <- dat$X.p
coords <- as.matrix(dat$coords)

# Package all data into a list
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = c(2, 2),
                   phi.unif = c(3/1, 3/.1))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
```

```
                          phi = 3 / .5,
                          sigma.sq = 2,
                          w = rep(0, nrow(X)),
                          z = apply(y, 1, max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- spPGOcc(occ.formula = ~ occ.cov,
                det.formula = ~ det.cov.1 + det.cov.2,
                data = data.list,
                inits = inits.list,
                n.batch = n.batch,
                batch.length = batch.length,
                priors = prior.list,
                cov.model = "exponential",
                tuning = tuning.list,
                NNGP = FALSE,
                n.neighbors = 5,
                search.type = 'cb',
                n.report = 10,
                n.burn = 50,
                n.chains = 1)

summary(out)
```

---

stMsPGOcc	*Function for Fitting Multi-Species Multi-Season Spatial Occupancy Models*

Description

The function `stMsPGOcc` fits multi-species multi-season spatial occupancy models with species correlations (i.e., a spatially-explicit joint species distribution model with imperfect detection). We use Polya-Gamma latent variables and a spatial factor modeling approach. Models are implemented using a Nearest Neighbor Gaussian Process.

Usage

```
stMsPGOcc(occ.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, n.omp.threads = 1,
          verbose = TRUE, ar1 = FALSE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
          n.chains = 1, ...)
```

Arguments

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

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 are allowed using lme4 syntax (Bates et al. 2015).

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, and coords. y is a four-dimensional array with first dimension equal to the number of species, second dimension equal to the number of sites, third dimension equal to the number of primary time periods, and fourth dimension equal to the maximum number of secondary replicates at a given site. occ.covs is a list of variables included in the occurrence portion of the model. Each list element is a different occurrence covariate, which can be site level or site/primary time period level. Site-level covariates are specified as a vector of length J while site/primary time period level covariates are specified as a matrix with rows corresponding to sites and columns correspond to primary time periods. Similarly, det.covs is a list of variables included in the detection portion of the model, with each list element corresponding to an individual variable. In addition to site-level and/or site/primary time period-level, detection covariates can also be observational-level. Observation-level covariates are specified as a three-dimensional array with first dimension corresponding to sites, second dimension corresponding to primary time period, and third dimension corresponding to replicate. coords is a $J \times 2$ matrix of the observation coordinates. Note that spOccupancy 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.psi, sigma.sq.p, z, phi, lambda, nu, sigma.sq.t, and rho. nu is only specified if cov.model = "matern", and sigma.sq.psi and sigma.sq.p are only specified if random effects are included in occ.formula or det.formula, respectively. sigma.sq.t and rho are only relevant when ar1 = TRUE. 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.psi, sigma.sq.p, phi.unif, nu.unif, sigma.sq.t.ig, and rho.unif. Community-level occurrence (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 set to 2.73. By default, community-level variance parameters for occupancy (`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 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 factor combinations. 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 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.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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. parameters are set to 0.1. `sigma.sq.t` and `rho` are the AR(1) variance and correlation parameters for the AR(1) zero-mean temporal random effects, respectively. `sigma.sq.t` is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which can each be specified as vector equal to the number of species in the model or a single value if the same prior is used for all species. `rho` is assumed to follow a uniform distribution, where the hyperparameters are specified similarly as a list of length two with the first and second elements corresponding to the lower and upper bounds of the uniform prior, which can each be specified as vector equal to the number of species in the model or a single value if the same prior is used for all species.

tuning   a list with each tag corresponding to a parameter name. Valid tags are `phi`, `nu`, `rho`. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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. If `FALSE`, a full Gaussian process is used. See Datta et al. (2016) and Finley et al. (2019) for more information. 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 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.
`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.
`ar1`	logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If `FALSE`, the model is fit without an AR(1) temporal autocovariance structure. If `TRUE`, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.
`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 stMsPGOcc that is a list comprised of:

beta.comm.samples

a coda object of posterior samples for the community level occurrence 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 occurrence 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 occurrence regression coefficients.

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

theta.samples   a coda object of posterior samples for the species level correlation parameters and the species-level temporal autocorrelation parameters.

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

z.samples   a four-dimensional array of posterior samples for the latent occurrence values for each species. Dimensions corresopnd to MCMC sample, species, site, and primary time period.

psi.samples   a four-dimensional array of posterior samples for the latent occupancy probability values for each species. Dimensions correspond to MCMC sample, species, site, and primary time period.

w.samples   a three-dimensional array of posterior samples for the latent spatial random effects for each spatial factor. Dimensions correspond to MCMC sample, factor, and site.

sigma.sq.psi.samples

a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

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

like.samples   a four-dimensional array of posterior samples for the likelihood value used for
                    calculating WAIC. Dimensions correspond to MCMC sample, species, site, and
                    time period.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribu-
tion is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code
implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.
lib.utexas.edu/handle/2152/21842.

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 Gaus-
sian process models for large geostatistical datasets. *Journal of the American Statistical Associa-
tion*, doi:10.1080/01621459.2015.1044091.

Finley, A.O., A. Datta, B.D. Cook, D.C. Morton, H.E. Andersen, and S. Banerjee. (2019) Effi-
cient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational
and Graphical Statistics*, doi:10.1080/10618600.2018.1537924.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian
process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using
Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists.
Ecological Monographs, 85(1), 3-28.

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

```
# Simulate Data -------------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
}
N <- 7
# Community-level covariate effects
# Occurrence
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
svc.cols <- c(1)
p.svc <- length(svc.cols)
n.factors <- 3
phi <- runif(p.svc * n.factors, 3 / .9, 3 / .3)
factor.model <- TRUE
cov.model <- 'exponential'
ar1 <- TRUE
sigma.sq.t <- runif(N, 0.05, 1)
rho <- runif(N, 0.1, 1)

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
                 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
```

```
                         psi.RE = psi.RE, p.RE = p.RE, factor.model = factor.model,
                         svc.cols = svc.cols, n.factors = n.factors, phi = phi, sp = sp,
                         cov.model = cov.model, ar1 = ar1, sigma.sq.t = sigma.sq.t, rho = rho)

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

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

data.list <- list(y = y, occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
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),
                   rho.unif = list(a = -1, b = 1),
                   sigma.sq.t.ig = list(a = 0.1, b = 0.1),
                   phi.unif = list(a = 3 / .9, b = 3 / .1))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
                   rho = 0.5, sigma.sq.t = 0.5,
                   phi = 3 / .5, z = z.init)
# Tuning
tuning.list <- list(phi = 1, rho = 0.5)

# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

out <- stMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2,
                 det.formula = ~ det.cov.1 + det.cov.2,
                 data = data.list,
                 inits = inits.list,
                 n.batch = n.batch,
                 batch.length = batch.length,
                 accept.rate = 0.43,
                 ar1 = TRUE,
                 NNGP = TRUE,
                 n.neighbors = 5,
```

```
                          n.factors = n.factors,
                          cov.model = 'exponential',
                          priors = prior.list,
                          tuning = tuning.list,
                          n.omp.threads = 1,
                          verbose = TRUE,
                          n.report = 1,
                          n.burn = n.burn,
                          n.thin = n.thin,
                          n.chains = 1)

     summary(out)
```

---

stPGOcc   *Function for Fitting Multi-Season Single-Species Spatial Occupancy*
                              Models Using Polya-Gamma Latent Variables

---

Description

Function for fitting multi-season single-species spatial occupancy models using Polya-Gamma latent variables.

Usage

```
stPGOcc(occ.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, n.omp.threads = 1,
        verbose = TRUE, ar1 = FALSE, n.report = 100,
        n.burn = round(.10 * n.batch * batch.length),
        n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
        k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, and coords. y is a three-dimensional array with first dimension equal to the number of sites (J), second dimension equal to the maximum number of primary time periods (i.e., years or seasons), and third dimension equal to the

maximum number of replicates at a given site. `occ.covs` is a list of variables included in the occurrence portion of the model. Each list element is a different occurrence covariate, which can be site level or site/primary timer period level. Site-level covariates are specified as a vector of length J while site/primary time period level covariates are specified as a matrix with rows corresponding to sites and columns correspond to primary time periods. Similarly, `det.covs` is a list of variables included in the detection portion of the model, with each list element corresponding to an individual variable. In addition to site-level and/or site/primary time period-level, detection covariates can also be observational-level. Observation-level covariates are specified as a three-dimensional array with first dimension corresponding to sites, second dimension corresponding to primary time period, and third dimension corresponding to replicate. `coords` is a $J \times 2$ matrix of the observation coordinates. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

inits   a list with each tag corresponding to a parameter name. Valid tags are `z`, `beta`, `alpha`, `sigma.sq`, `phi`, `w`, `nu`, `sigma.sq.psi`, `sigma.sq.p`, `sigma.sq.t`, `rho`. The value portion of each tag is the parameter's initial value. `sigma.sq.psi` and `sigma.sq.p` are only relevant when including random effects in the occurrence and detection portion of the occupancy model, respectively. `nu` is only specified if `cov.model = "matern"`. `sigma.sq.t` and `rho` are only relevant when `ar1 = TRUE`. 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`, `sigma.sq.psi.ig`, `sigma.sq.p.ig`, `phi.unif`, `sigma.sq.ig`, `nu.unif`, `sigma.sq.t.ig`, and `rho.unif`. Occupancy (`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 2.72. `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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. The spatial variance parameter, `sigma.sq`, is assumed to follow an inverse-Gamma distribution. The spatial decay `phi` and smoothness `nu` parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for `sigma.sq.ig` are passed as a vector of length two, with the first and second elements corresponding to the shape and scale parameters, 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.t` and `rho` are the AR(1) variance and correlation parameters

for the AR(1) zero-mean temporal random effects, respectively. `sigma.sq.t` is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a vector with elements corresponding to the shape and scale parameters, respectively. `rho` is assumed to follow a uniform distribution, where the hyperparameters are specified in a vector of length two with elements corresponding to the lower and upper bounds of the uniform prior.

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. Valid tags are `phi`, `nu`, and `rho`. The value portion of each tag defines the initial variance of the Adaptive sampler. See Roberts and Rosenthal (2009) for details.

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. Currently only `NNGP = TRUE` is supported for multi-season single-species trend occupancy models.

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 or k-fold cross-validation.

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.

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. 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.

ar1   logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If `FALSE`, the model is fit without an AR(1) temporal

autocovariance structure. If `TRUE`, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.

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.

k.fold   specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and `k.fold.threads` and `k.fold.seed` are ignored. In *k*-fold cross-validation, the data specified in `data` is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). For cross-validation in multi-season models, the data are split along the site dimension, such that each hold-out data set consists of a `J / k.fold` sites sampled over all primary time periods during which data are available at each given site. Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the `k.fold.deviance` object in the return list.

k.fold.threads number of threads to use for cross-validation. If `k.fold.threads > 1` parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if `k.fold` is not specified.

k.fold.seed   seed used to split data set into `k.fold` parts for k-fold cross-validation. Ignored if `k.fold` is not specified.

k.fold.only   a logical value indicating whether to only perform cross-validation (`TRUE`) or perform cross-validation after fitting the full model (`FALSE`). Default value is `FALSE`.

...   currently no additional arguments

Value

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

beta.samples   a coda object of posterior samples for the occupancy regression coefficients.

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

z.samples   a three-dimensional array of posterior samples for the latent occupancy values, with dimensions corresponding to posterior sample, site, and primary time period.

psi.samples   a three-dimensional array of posterior samples for the latent occupancy probability values, with dimensions corresponding to posterior sample, site, and primary time period.

theta.samples   a coda object of posterior samples for spatial covariance parameters and temporal covariance parameters if `ar1 = TRUE`.

w.samples   a coda object of posterior samples for latent spatial random effects.

sigma.sq.psi.samples

a coda object of posterior samples for variances of random intercepts included in the occupancy portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

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

eta.samples   a coda object of posterior samples for the AR(1) random effects for each primary time period. Only included if ar1 = TRUE

.

like.samples   a three-dimensional array of posterior samples for the likelihood values associated with each site and primary time period. 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().

k.fold.deviance

scoring rule (deviance) from k-fold cross-validation. Only included if k.fold is specified in function call.

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(). Note that if k.fold.only = TRUE, the return list object will only contain run.time and k.fold.deviance.

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Kery, M., & Royle, J. A. (2021). Applied hierarchical modeling in ecology: Analysis of distribution, abundance and species richness in R and BUGS: Volume 2: Dynamic and advanced models. Academic Press. Section 4.6.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm. 2002. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology 83: 2248-2255.

Examples

```
set.seed(500)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Primary time periods
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence -------------------------
beta <- c(0.4, 0.5, -0.9)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection --------------------------
alpha <- c(-1, 0.7, -0.5)
p.RE <- list()
# Spatial ----------------------------
sp <- TRUE
cov.model <- "exponential"
sigma.sq <- 2
phi <- 3 / .4
# Temporal ---------------------------
rho <- 0.5
sigma.sq.t <- 1

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
               beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
               psi.RE = psi.RE, p.RE = p.RE, sp = TRUE, sigma.sq = sigma.sq,
```

```
                      phi = phi, cov.model = cov.model, ar1 = TRUE,
                      sigma.sq.t = sigma.sq.t, rho = rho)

# Package all data into a list
# Occurrence
occ.covs <- list(int = dat$X[, , 1],
                 trend = dat$X[, , 2],
                 occ.cov.1 = dat$X[, , 3])
# Detection
det.covs <- list(det.cov.1 = dat$X.p[, , , 2],
                 det.cov.2 = dat$X.p[, , , 3])
# Data list bundle
data.list <- list(y = dat$y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = dat$coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = c(2, 2),
                   phi.unif = c(3 / 1, 3 / 0.1),
                   rho.unif = c(-1, 1),
                   sigma.sq.t.ig = c(2, 1))

# Initial values
z.init <- apply(dat$y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0, z = z.init, phi = 3 / .5, sigma.sq = 2,
                   w = rep(0, J), rho = 0, sigma.sq.t = 0.5)
# Tuning
tuning.list <- list(phi = 1, rho = 1)
# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length

# Run the model
out <- stPGOcc(occ.formula = ~ trend + occ.cov.1,
               det.formula = ~ det.cov.1 + det.cov.2,
               data = data.list,
               inits = inits.list,
               n.batch = n.batch,
               batch.length = batch.length,
               priors = prior.list,
               cov.model = "exponential",
               tuning = tuning.list,
               NNGP = TRUE,
               ar1 = TRUE,
               n.neighbors = 5,
               search.type = 'cb',
               n.report = 10,
               n.burn = 50,
               n.chains = 1)
```

```
summary(out)
```

summary.intMsPGOcc   *Methods for intMsPGOcc Object*

Description

Methods for extracting information from fitted integrated multi-species occupancy (intMsPGOcc) models.

Usage

```
## S3 method for class 'intMsPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'intMsPGOcc'
print(x, ...)
## S3 method for class 'intMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class intMsPGOcc.
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.comm", "tau.sq.beta", "alpha", "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 intMsPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a intMsPGOcc object.

summary.intPGOcc *Methods for intPGOcc Object*

Description

Methods for extracting information from fitted single species integrated occupancy (intPGOcc) model.

Usage

```
## S3 method for class 'intPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'intPGOcc'
print(x, ...)
## S3 method for class 'intPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class intPGOcc.
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", "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 intPGOcc, including methods to the generic functions [print](), [summary](), and [plot]().

Value

No return value, called to display summary information of a intPGOcc object.

---

summary.lfJSDM   *Methods for lfJSDM Object*

Description

Methods for extracting information from a fitted latent factor joint species distribution model (lfJSDM).

Usage

```
## S3 method for class 'lfJSDM'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'lfJSDM'
print(x, ...)
## S3 method for class 'lfJSDM'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class lfJSDM.
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.psi", "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 lfJSDM, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a lfJSDM object.

---

summary.lfMsPGOcc   *Methods for lfMsPGOcc Object*

Description

Methods for extracting information from a fitted latent factor multi-species occupancy model (`lfMsPGOcc`).

Usage

```
## S3 method for class 'lfMsPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'lfMsPGOcc'
print(x, ...)
## S3 method for class 'lfMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class `lfMsPGOcc`.
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.psi"`, `"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 `lfMsPGOcc`, including methods to the generic functions `print`, `summary`, and `plot`.

Value

No return value, called to display summary information of a `lfMsPGOcc` object.

---

summary.msPGOcc   *Methods for msPGOcc Object*

Description

Methods for extracting information from fitted multi-species occupancy (msPGOcc) model.

Usage

```
## S3 method for class 'msPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'msPGOcc'
print(x, ...)
## S3 method for class 'msPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class msPGOcc.
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.psi", "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 msPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a msPGOcc object.

---

summary.PGOcc   *Methods for PGOcc Object*

Description

Methods for extracting information from fitted single-species occupancy (PGOcc) model.

Usage

```
## S3 method for class 'PGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'PGOcc'
print(x, ...)
## S3 method for class 'PGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class PGOcc.
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.psi", "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 PGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a PGOcc object.

---

summary.postHocLM   *Methods for postHocLM Object*

Description

Methods for extracting information from fitted posthoc linear models (postHocLM).

Usage

```
## S3 method for class 'postHocLM'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'postHocLM'
print(x, ...)
```

Arguments

object, x	object of class postHocLM.
quantiles	for summary, posterior distribution quantiles to compute.
digits	for summary, number of digits to report.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted model objects of class postHocLM, including methods to the generic functions print and summary.

Value

No return value, called to display summary information of a postHocLM object.

---

summary.ppcOcc   *Methods for ppcOcc Object*

Description

Methods for extracting information from posterior predictive check objects of class ppcOcc.

Usage

```
## S3 method for class 'ppcOcc'
summary(object, level, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object	object of class ppcOcc.
level	a quoted keyword for multi-species models that indicates the level to summarize the posterior predictive check. Valid key words are: "community", "species", or "both".
digits	number of digits to report.
...	currently no additional arguments

Details

A set of standard extractor functions for fitted posterior predictive check objects of class ppcOcc, including methods to the generic function summary.

Value

No return value, called to display summary information of a ppcOcc object.

---

summary.sfJSDM   *Methods for sfJSDM Object*

Description

Methods for extracting information from fitted spatial factor joint species distribution models (sfJSDM).

Usage

```
## S3 method for class 'sfJSDM'
summary(object, level, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'sfJSDM'
print(x, ...)
## S3 method for class 'sfJSDM'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class sfJSDM.
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.psi", "beta.comm", "tau.sq.beta", "theta", "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 sfJSDM, including methods to the generic functions [print](), [summary](), and [plot]().

Value

No return value, called to display summary information of a sfJSDM object.

---

summary.sfMsPGOcc   *Methods for sfMsPGOcc Object*

Description

Methods for extracting information from fitted spatial factor multi-species occupancy model.

Usage

```
## S3 method for class 'sfMsPGOcc'
summary(object, level, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'sfMsPGOcc'
print(x, ...)
## S3 method for class 'sfMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class sfMsPGOcc.
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.psi", "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 sfMsPGOcc, including methods to the generic functions [print](), [summary](), and [plot]().

Value

No return value, called to display summary information of a sfMsPGOcc object.

summary.spIntPGOcc *Methods for spIntPGOcc Object*

Description

Methods for extracting information from fitted single-species spatial integrated occupancy (spIntPGOcc) model.

Usage

```
## S3 method for class 'spIntPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spIntPGOcc'
print(x, ...)
## S3 method for class 'spIntPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spIntPGOcc.
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", "alpha", "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 spIntPGOcc, including methods to the generic functions [print](), [summary](), and [plot]().

Value

No return value, called to display summary information of a spIntPGOcc object.

summary.spMsPGOcc   *Methods for spMsPGOcc Object*

Description

Methods for extracting information from fitted multi-species spatial occupancy (spMsPGOcc) model.

Usage

```
## S3 method for class 'spMsPGOcc'
summary(object, level, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spMsPGOcc'
print(x, ...)
## S3 method for class 'spMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spMsPGOcc.
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.psi", "beta.comm", "tau.sq.beta", "alpha", "alpha.star", "sigma.sq.p", "alpha.comm", "tau.sq.alpha", "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 spMsPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a spMsPGOcc object.

---

summary.spPGOcc *Methods for spPGOcc Object*

Description

Methods for extracting information from fitted single-species spatial occupancy (spPGOcc) model.

Usage

```
## S3 method for class 'spPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'spPGOcc'
print(x, ...)
## S3 method for class 'spPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class spPGOcc.
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.psi", "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 spPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a spPGOcc object.

---

summary.stMsPGOcc   *Methods for stMsPGOcc Object*

Description

Methods for extracting information from fitted multi-species, multi-season spatial occupancy (`stMsPGOcc`) model.

Usage

```
## S3 method for class 'stMsPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'stMsPGOcc'
print(x, ...)
## S3 method for class 'stMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

`object, x`	object of class `stMsPGOcc`.
`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.psi"`, `"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 `stMsPGOcc`, including methods to the generic functions `print`, `summary`, and `plot`.

Value

No return value, called to display summary information of a `stMsPGOcc` object.

summary.stPGOcc *Methods for stPGOcc Object*

Description

Methods for extracting information from fitted multi-season single-species spatial occupancy (stPGOcc) model.

Usage

```
## S3 method for class 'stPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'stPGOcc'
print(x, ...)
## S3 method for class 'stPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class stPGOcc.
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.psi", "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 stPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a stPGOcc object.

summary.svcMsPGOcc   *Methods for svcMsPGOcc Object*

Description

Methods for extracting information from fitted multi-species spatially-varying coefficient occupancy model.

Usage

```
## S3 method for class 'svcMsPGOcc'
summary(object, level, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcMsPGOcc'
print(x, ...)
## S3 method for class 'svcMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcMsPGOcc.
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.psi", "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 svcMsPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcMsPGOcc object.

summary.svcPGBinom   *Methods for svcPGBinom Object*

Description

Methods for extracting information from fitted single-species spatially-varying coefficient binomial model (svcPGBinom).

Usage

```
## S3 method for class 'svcPGBinom'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcPGBinom'
print(x, ...)
## S3 method for class 'svcPGBinom'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcPGBinom.
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.psi", "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 svcPGBinom, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcPGBinom object.

summary.svcPGOcc   *Methods for svcPGOcc Object*

Description

Methods for extracting information from fitted single-species spatially-varying coefficient occupancy (svcPGOcc) model.

Usage

```
## S3 method for class 'svcPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcPGOcc'
print(x, ...)
## S3 method for class 'svcPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcPGOcc.
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.psi", "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 svcPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcPGOcc object.

summary.svcTMsPGOcc   *Methods for svcTMsPGOcc Object*

Description

Methods for extracting information from fitted multi-species, multi-season spatially-varying coefficient occupancy (svcTMsPGOcc) model.

Usage

```
## S3 method for class 'svcTMsPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcTMsPGOcc'
print(x, ...)
## S3 method for class 'svcTMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcTMsPGOcc.
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.psi", "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 svcTMsPGOcc, including methods to the generic functions [print](), [summary](), and [plot]().

Value

No return value, called to display summary information of a svcTMsPGOcc object.

summary.svcTPGBinom   *Methods for svcTPGBinom Object*

Description

Methods for extracting information from fitted multi-season single-species spatially-varying coefficient binomial model (`svcTPGBinom`).

Usage

```
## S3 method for class 'svcTPGBinom'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcTPGBinom'
print(x, ...)
## S3 method for class 'svcTPGBinom'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcTPGBinom.
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.psi", "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 svcTPGBinom, including methods to the generic functions `print`, `summary`, `plot`.

Value

No return value, called to display summary information of a svcTPGBinom object.

summary.svcTPGOcc   *Methods for svcTPGOcc Object*

Description

Methods for extracting information from fitted multi-season single-species spatially-varying coefficient occupancy (svcTPGOcc) model.

Usage

```
## S3 method for class 'svcTPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'svcTPGOcc'
print(x, ...)
## S3 method for class 'svcTPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class svcTPGOcc.
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.psi", "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 svcTPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a svcTPGOcc object.

summary.tMsPGOcc   *Methods for tMsPGOcc Object*

Description

Methods for extracting information from fitted multi-species, multi-season occupancy (tMsPGOcc) model.

Usage

```
## S3 method for class 'tMsPGOcc'
summary(object, level = 'both', quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'tMsPGOcc'
print(x, ...)
## S3 method for class 'tMsPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class tMsPGOcc.
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.psi″, ″beta.comm″, ″tau.sq.beta″, ″alpha″, ″alpha.star″, ″sigma.sq.p″, ″alpha.comm″, ″tau.sq.alpha″, ″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 tMsPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a tMsPGOcc object.

summary.tPGOcc *Methods for tPGOcc Object*

Description

Methods for extracting information from fitted multi-season single-species occupancy (tPGOcc) model.

Usage

```
## S3 method for class 'tPGOcc'
summary(object, quantiles = c(0.025, 0.5, 0.975),
        digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'tPGOcc'
print(x, ...)
## S3 method for class 'tPGOcc'
plot(x, param, density = TRUE, ...)
```

Arguments

object, x	object of class tPGOcc.
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.psi", "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 tPGOcc, including methods to the generic functions print, summary, and plot.

Value

No return value, called to display summary information of a tPGOcc object.

---

svcMsPGOcc   *Function for Fitting Multi-Species Spatially-Varying Coefficient Oc-*
                                    cupancy Models

Description

The function svcMsPGOcc fits multi-species spatially-varying coefficient occupancy models with
species correlations (i.e., a spatially-explicit joint species distribution model with imperfect detec-
tion). We use Polya-Gamma latent variables and a spatial factor modeling approach. Models are
implemented using a Nearest Neighbor Gaussian Process.

Usage

```
svcMsPGOcc(occ.formula, det.formula, data, inits, priors, tuning,
           svc.cols = 1, cov.model = 'exponential', NNGP = TRUE,
           n.neighbors = 15, search.type = 'cb', std.by.sp = FALSE,
           n.factors, n.batch, batch.length,
           accept.rate = 0.43, n.omp.threads = 1,
           verbose = TRUE, n.report = 100,
           n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
           n.chains = 1, ...)
```

Arguments

occ.formula	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Random intercepts are allowed using lme4 syntax (Bates et al. 2015). Only right-hand side of formula is specified. See example below.
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 are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, occ.covs, det.covs, coords, and range.ind. y is a three-dimensional array 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. occ.covs is a matrix or data frame containing the variables used in the occurrence 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 spOccupancy assumes coordinates are specified in a projected coordinate system. range.ind is a matrix with rows corresponding to

species and columns corresponding to sites, with each element taking value 1 if that site is within the range of the corresponding species and 0 if it is outside of the range. This matrix is not required, but it can be helpful to restrict the modeled area for each individual species to be within the realistic range of locations for that species when estimating the model parameters. This is applicable when auxiliary data sources are available on the realistic range of the species.

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.psi`, `sigma.sq.p`, `z`, `phi`, `lambda`, and `nu`. `nu` is only specified if `cov.model = "matern"`, and `sigma.sq.psi` and `sigma.sq.p` are only specified if random effects are included in `occ.formula` or `det.formula`, respectively. 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.psi`, `sigma.sq.p`, `phi.unif`, and `nu.unif`. Community-level occurrence (`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 set to 2.72. By default, community-level variance parameters for occupancy (`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 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 for each spatially-varying coefficient specified in `svc.cols`. 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 * length(svc.cols)` corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factor/SVC combinations. The priors for the factor loadings matrix `lambda` for each SVC are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen and Amemlya 2002). The upper triangular elements of the `N x n.factors` matrix are fixed at 0 and the diagonal elements are fixed at 1 for each SVC. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1). `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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 pa-

	rameters, 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.
tuning	a list with each tag corresponding to a parameter name. Valid tags are phi and nu. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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 occupancy model analogous to sfMsPGOcc (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. 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.
std.by.sp	a logical value indicating whether the covariates are standardized separately for each species within the corresponding range for each species (TRUE) or not (FALSE). Note that if range.ind is specified in data.list, this will result in the covariates being standardized differently for each species based on the sites where range.ind == 1 for that given species. If range.ind is not specified and std.by.sp = TRUE, this will simply be equivalent to standardizing the covariates across all locations prior to fitting the model.
n.factors	the number of factors to use in the spatial factor model approach. Note this corresponds to the number of factors used for each spatially-varying coefficient that is estimated in the model. 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.

n.omp.threads   a positive integer indicating the number of threads to use for SMP parallel pro-
                 cessing. 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 sys-
                 tems.

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 svcMsPGOcc that is a list comprised of:

beta.comm.samples
                 a coda object of posterior samples for the community level occurrence regres-
                 sion 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 occurrence community variance pa-
                 rameters.
tau.sq.alpha.samples
                 a coda object of posterior samples for the detection community variance param-
                 eters.
beta.samples   a coda object of posterior samples for the species level occurrence regression
                 coefficients.
alpha.samples   a coda object of posterior samples for the species level detection regression
                 coefficients.
theta.samples   a coda object of posterior samples for the species level correlation parameters
                 for each spatially-varying coefficient.

lambda.samples	a coda object of posterior samples for the latent spatial factor loadings for each spatially-varying coefficient.
z.samples	a three-dimensional array of posterior samples for the latent occurrence values for each species.
psi.samples	a three-dimensional array of posterior samples for the latent occupancy probability values for each species.
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.psi.samples	
	a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.
sigma.sq.p.samples	
	a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.
beta.star.samples	
	a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.
alpha.star.samples	
	a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in det.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. Note that detection probability estimated values are not included in the model object, but can be extracted using fitted().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

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(400)

# Simulate Data -------------------------------------------------------
J.x <- 10
J.y <- 10
J <- J.x * J.y
n.rep <- sample(5, size = J, replace = TRUE)
N <- 6
# Community-level covariate effects
# Occurrence
beta.mean <- c(0.2, -0.2, 0.3, -0.1, 0.4)
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 0.4, 0.5, 0.3)
# Detection
alpha.mean <- c(0, 1.2, -0.5)
tau.sq.alpha <- c(1, 0.5, 1.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list(levels = 15,
               sigma.sq.psi = 0.7)
p.RE <- list(levels = 20,
             sigma.sq.p = 0.5)
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
```

```
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
# Number of spatial factors for each SVC
n.factors <- 2
# The intercept and first two covariates have spatially-varying effects
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
q.p.svc <- n.factors * p.svc
# Spatial decay parameters
phi <- runif(q.p.svc, 3 / 0.9, 3 / 0.1)
# A length N vector indicating the proportion of simulated locations
# that are within the range for a given species.
range.probs <- runif(N, 0.4, 1)
factor.model <- TRUE
cov.model <- 'spherical'
sp <- TRUE

dat <- simMsOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, N = N, beta = beta, alpha = alpha,
                psi.RE = psi.RE, p.RE = p.RE, phi = phi, sp = sp, svc.cols = svc.cols,
                cov.model = cov.model, n.factors = n.factors,
                factor.model = factor.model, range.probs = range.probs)

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
range.ind <- dat$range.ind

# Prep data for spOccupancy ----------------------------------------------
# Occurrence covariates
occ.covs <- cbind(X, X.re)
colnames(occ.covs) <- c('int', 'occ.cov.1', 'occ.cov.2', 'occ.cov.3',
                        'occ.cov.4', 'occ.factor.1')
# Detection covariates
det.covs <- list(det.cov.1 = X.p[, , 2],
                 det.cov.2 = X.p[, , 3],
                 det.factor.1 = X.p.re[, , 1])
# Data list
data.list <- list(y = y, coords = coords, occ.covs = occ.covs,
                  det.covs = det.covs, range.ind = range.ind)
# Priors
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, b = 3 / .1))
inits.list <- list(alpha.comm = 0,
                   beta.comm = 0,
```

```
                          beta = 0,
                          alpha = 0,
                          tau.sq.beta = 1,
                          tau.sq.alpha = 1,
                          z = apply(y, c(1, 2), max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

# Number of batches
n.batch <- 2
# Batch length
batch.length <- 25
n.burn <- 0
n.thin <- 1
n.samples <- n.batch * batch.length

out <- svcMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2 + occ.cov.3 +
                                  occ.cov.4 + (1 | occ.factor.1),
                  det.formula = ~ det.cov.1 + det.cov.2 + (1 | det.factor.1),
                  data = data.list,
                  inits = inits.list,
                  n.batch = n.batch,
                  n.factors = n.factors,
                  batch.length = batch.length,
                  std.by.sp = TRUE,
                  accept.rate = 0.43,
                  priors = prior.list,
                  svc.cols = svc.cols,
                  cov.model = "spherical",
                  tuning = tuning.list,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  NNGP = TRUE,
                  n.neighbors = 5,
                  search.type = 'cb',
                  n.report = 10,
                  n.burn = n.burn,
                  n.thin = n.thin,
                  n.chains = 1)

  summary(out)
```

---

svcPGBinom   *Function for Fitting Single-Species Spatially-Varying Coefficient Bi-*
                               nomial Models Using Polya-Gamma Latent Variables

---

Description

The function `svcPGBinom` fits single-species spatially-varying coefficient binomial models using
Polya-Gamma latent variables. Models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
svcPGBinom(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,
           n.omp.threads = 1, verbose = TRUE, n.report = 100,
           n.burn = round(.10 * n.batch * batch.length),
           n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
           k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data
: a list containing data necessary for model fitting. Valid tags are y, covs, weights, and coords. y is a numeric vector containing the binomial data with length equal to the total number of sites (J). covs is a matrix or data frame containing the covariates used in the model, with J rows for each column (variable). weights is a numeric vector containing the binomial weights (i.e., the total number of Bernoulli trials) at each site. If weights is not specified, svcPGBinom assumes 1 trial at each site (i.e., presence/absence). coords is a $J \times 2$ matrix of the observation coordinates. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

inits
: a list with each tag corresponding to a parameter name. Valid tags are beta, sigma.sq, phi, w, nu, and sigma.sq.psi. nu is only specified if cov.model = "matern", and sigma.sq.psi is only specified if there are random effects 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.normal, phi.unif, sigma.sq.ig, sigma.sq.unif, nu.unif, and sigma.sq.psi.ig. Regression coefficients (beta) 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 2.73. The spatial variance parameter, sigma.sq, for each spatially-varying coefficient is assumed to follow an inverse-Gamma distribution or a uniform distribution (default is inverse-Gamma). The spatial decay phi and smoothness nu parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for sigma.sq are passed as a list with two elements corresponding to the shape and scale parameters, respetively, with each element comprised of a vector equal to the number of spatially-varying coefficients to be estimated or of length one if priors are the same for all coefficients. The

hyperparameters of any uniform priors are also passed as a list of length two with the first and second elements corresponding to the lower and upper support, respectively, which can be passed as a vector equal to the total number of spatially-varying coefficients to be estimated or of length one if priors are the same for all coefficients. `sigma.sq.psi` are the random effect variances for any 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 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 `covs` (for the intercept, use '`(Intercept)`').

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. Valid tags are `phi`, `sigma.sq`, and `nu`. The value portion of each tag defines the initial variance of the Adaptive sampler. See Roberts and Rosenthal (2009) for details.

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.

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 or k-fold cross-validation.

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.

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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the regression coefficients.
y.rep.samples	a coda object of posterior samples for the fitted data values
psi.samples	a coda object of posterior samples for the occurrence probability values
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a three-dimensional array of posterior samples for the latent spatial random effects for all spatially-varying coefficients. Dimensions correspond to MCMC sample, coefficient, and sites.
sigma.sq.psi.samples	
	a coda object of posterior samples for variances of unstructured random intercepts included in the model. Only included if random intercepts are specified in formula.

beta.star.samples

> a coda object of posterior samples for the unstructured random effects. Only included if random intercepts 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()`.

k.fold.deviance

> soring rule (deviance) from k-fold cross-validation. Only included if `k.fold` is specified in function call.

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.

Finley, A. O., and Banerjee, S. (2020). Bayesian spatially varying coefficient models in the spBayes R package. *Environmental Modelling and Software*, 125, 104608.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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
# Binomial weights
weights <- sample(10, J, replace = TRUE)
beta <- c(0, 0.5, -0.2, 0.75)
```

```
p <- length(beta)
# No unstructured random effects
psi.RE <- list()
# Spatial parameters
sp <- TRUE
# Two spatially-varying covariates.
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.4, 1.5)
phi <- runif(p.svc, 3/1, 3/0.2)

# Simulate the data
dat <- simBinom(J.x = J.x, J.y = J.y, weights = weights, beta = beta,
                psi.RE = psi.RE, sp = sp, svc.cols = svc.cols,
                cov.model = cov.model, sigma.sq = sigma.sq, phi = phi)

# Binomial data
y <- dat$y
# Covariates
X <- dat$X
# Spatial coordinates
coords <- dat$coords

# Package all data into a list
# Covariates
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,
                  weights = weights)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   sigma.sq.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)
# Tuning
tuning.list <- list(phi = 1)

n.batch <- 10
batch.length <- 25
n.burn <- 100
n.thin <- 1

out <- svcPGBinom(formula = ~ cov.1 + cov.2 + cov.3,
                  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,
                         cov.model = "exponential",
                         tuning = tuning.list,
                         n.omp.threads = 1,
                         verbose = TRUE,
                         NNGP = TRUE,
                         n.neighbors = 5,
                         n.report = 2,
                         n.burn = n.burn,
                         n.thin = n.thin,
                         n.chains = 1)

summary(out)
```

---

svcPGOcc   *Function for Fitting Single-Species Spatially-Varying Coefficient Oc-*
                           cupancy Models Using Polya-Gamma Latent Variables

---

Description

The function svcPGOcc fits single-species spatially-varying coefficient occupancy models using
Polya-Gamma latent variables. Models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
svcPGOcc(occ.formula, det.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,
         n.omp.threads = 1, verbose = TRUE, n.report = 100,
         n.burn = round(.10 * n.batch * batch.length),
         n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
         k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
                        det.covs, and coords. y is the detection-nondetection data matrix or data
                        frame with first dimension equal to the number of sites (J) and second dimen-
                        sion equal to the maximum number of replicates at a given site. occ.covs is
                        a matrix or data frame containing the variables used in the occupancy 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 spOccupancy assumes coordinates are specified in a projected coordinate
                        system.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta,
                        alpha, sigma.sq, phi, w, nu, sigma.sq.psi, sigma.sq.p. nu is only specified
                        if cov.model = "matern", sigma.sq.p is only specified if there are random ef-
                        fects in det.formula, and sigma.sq.psi is only specified if there are random
                        effects in occ.formula. The value portion of each tag is the parameter's initial
                        value. See priors description for definition of each parameter name. Addition-
                        ally, 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, sigma.sq.unif, nu.unif, sigma.sq.psi.ig,
                        and sigma.sq.p.ig. Occurrence (beta) and detection (alpha) regression coef-
                        ficients 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 ele-
                        ments 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 speci-
                        fied, prior means are set to 0 and prior variances set to 2.73. The spatial variance
                        parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution or
                        a uniform distribution (default is inverse-Gamma). The spatial decay phi and
                        smoothness nu parameters are assumed to follow Uniform distributions. The hy-
                        perparameters of the inverse-Gamma for sigma.sq are passed as a list with two
                        elements corresponding to the shape and scale parameters, respetively, with each
                        element comprised of a vector equal to the number of spatially-varying coeffi-
                        cients to be estimated or of length one if priors are the same for all coefficients.
                        The hyperparameters of any uniform priors are also passed as a list of length
                        two with the first and second elements corresponding to the lower and upper
                        support, respectively, which can be passed as a vector equal to the total number
                        of spatially-varying coefficients to be estimated or of length one if priors are the
                        same for all coefficients. sigma.sq.psi and sigma.sq.p are the random effect
                        variances for any occurrence 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 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 spatial occupancy model analogous to spPGOcc (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 list with each tag corresponding to a parameter name. Valid tags are phi, nu, and sigma.sq. The value portion of each tag defines the initial variance of the Adaptive sampler. See Roberts and Rosenthal (2009) for details.

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. Only NNGP = TRUE is currently supported for spatially-varying coefficient models.

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 or k-fold cross-validation.

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.

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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the occurrence regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
z.samples	a coda object of posterior samples for the latent occurrence values
psi.samples	a coda object of posterior samples for the latent occurrence probability values
theta.samples	a coda object of posterior samples for spatial covariance parameters.
w.samples	a three-dimensional array of posterior samples for the latent spatial random effects for all spatially-varying coefficients. Dimensions correspond to MCMC sample, coefficient, and sites.
sigma.sq.psi.samples	
	a coda object of posterior samples for variances of random intercepts included in the occupancy portion of the model. Only included if random intercepts are specified in occ.formula.
sigma.sq.p.samples	
	a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

`beta.star.samples`

a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in `occ.formula`.

`alpha.star.samples`

a coda object of posterior samples for the detection random effects. Only included if random intercepts are specified in `det.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()`.

`k.fold.deviance`

soring rule (deviance) from k-fold cross-validation. Only included if `k.fold` is specified in function call.

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.

Finley, A. O., and Banerjee, S. (2020). Bayesian spatially varying coefficient models in the spBayes R package. *Environmental Modelling and Software*, 125, 104608.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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(400)
# Simulate Data -------------------------------------------------------
J.x <- 8
J.y <- 8
```

```
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, 2)
p.occ <- length(beta)
alpha <- c(0, 1)
p.det <- length(alpha)
phi <- c(3 / .6, 3 / .8)
sigma.sq <- c(1.2, 0.7)
svc.cols <- c(1, 2)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sigma.sq = sigma.sq, phi = phi, sp = TRUE, cov.model = 'exponential',
              svc.cols = svc.cols)
# Detection-nondetection data
y <- dat$y
# Occupancy covariates
X <- dat$X
# Detection covarites
X.p <- dat$X.p
# Spatial coordinates
coords <- dat$coords

# Package all data into a list
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov.1 = X.p[, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)

# Number of batches
n.batch <- 10
# Batch length
batch.length <- 25
n.iter <- n.batch * batch.length
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = list(a = 2, b = 1),
                   phi.unif = list(a = 3/1, b = 3/.1))
# Initial values
inits.list <- list(alpha = 0, beta = 0,
                   phi = 3 / .5,
                   sigma.sq = 2,
                   w = matrix(0, nrow = length(svc.cols), ncol = nrow(X)),
                   z = apply(y, 1, max, na.rm = TRUE))
# Tuning
tuning.list <- list(phi = 1)

out <- svcPGOcc(occ.formula = ~ occ.cov,
                det.formula = ~ det.cov.1,
                data = data.list,
                inits = inits.list,
```

```
                           n.batch = n.batch,
                           batch.length = batch.length,
                           accept.rate = 0.43,
                           priors = prior.list,
                           cov.model = 'exponential',
                           svc.cols = c(1, 2),
                           tuning = tuning.list,
                           n.omp.threads = 1,
                           verbose = TRUE,
                           NNGP = TRUE,
                           n.neighbors = 5,
                           search.type = 'cb',
                           n.report = 10,
                           n.burn = 50,
                           n.thin = 1)

    summary(out)
```

---

svcTMsPGOcc	*Function for Fitting Multi-Species Multi-Season Spatially-Varying Coefficient Occupancy Models*

Description

The function svcTMsPGOcc fits multi-species multi-season spatially-varying coefficient occupancy models with species correlations (i.e., a spatially-explicit joint species distribution model with imperfect detection). We use Polya-Gamma latent variables and a spatial factor modeling approach. Models are implemented using a Nearest Neighbor Gaussian Process.

Usage

```
svcTMsPGOcc(occ.formula, det.formula, data, inits, priors, tuning,
            svc.cols = 1, cov.model = 'exponential', NNGP = TRUE,
            n.neighbors = 15, search.type = 'cb', std.by.sp = FALSE,
            n.factors, svc.by.sp, n.batch, batch.length,
            accept.rate = 0.43, n.omp.threads = 1,
            verbose = TRUE, ar1 = FALSE, n.report = 100,
            n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
            n.chains = 1, ...)
```

Arguments

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

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 are allowed using lme4 syntax (Bates et
                 al. 2015).

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
                 det.covs, coords, and range.ind. y is a four-dimensional array with first di-
                 mension equal to the number of species, second dimension equal to the number
                 of sites, third dimension equal to the number of primary time periods, and fourth
                 dimension equal to the maximum number of secondary replicates at a given site.
                 occ.covs is a list of variables included in the occurrence portion of the model.
                 Each list element is a different occurrence covariate, which can be site level or
                 site/primary time period level. Site-level covariates are specified as a vector of
                 length J while site/primary time period level covariates are specified as a matrix
                 with rows corresponding to sites and columns correspond to primary time peri-
                 ods. Similarly, det.covs is a list of variables included in the detection portion
                 of the model, with each list element corresponding to an individual variable. In
                 addition to site-level and/or site/primary time period-level, detection covariates
                 can also be observational-level. Observation-level covariates are specified as a
                 three-dimensional array with first dimension corresponding to sites, second di-
                 mension corresponding to primary time period, and third dimension correspond-
                 ing to replicate. coords is a $J \times 2$ matrix of the observation coordinates. Note
                 that spOccupancy assumes coordinates are specified in a projected coordinate
                 system. range.ind is a matrix with rows corresponding to species and columns
                 corresponding to sites, with each element taking value 1 if that site is within
                 the range of the corresponding species and 0 if it is outside of the range. This
                 matrix is not required, but it can be helpful to restrict the modeled area for each
                 individual species to be within the realistic range of locations for that species
                 when estimating the model parameters. This is applicable when auxiliary data
                 sources are available on the realistic range of the species.

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.psi, sigma.sq.p,
                 z, phi, lambda, nu, sigma.sq.t, and rho. nu is only specified if cov.model =
                 "matern", sigma.sq.t and rho are only specified if ar1 = TRUE, and sigma.sq.psi
                 and sigma.sq.p are only specified if random effects are included in occ.formula
                 or det.formula, respectively. 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.psi, sigma.sq.p,
                 phi.unif, nu.unif, sigma.sq.t.ig, and rho.unif. Community-level occur-
                 rence (beta.comm) and detection (alpha.comm) regression coefficients are as-
                 sumed to follow a normal distribution. The hyperparameters of the normal dis-
                 tribution 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 2.73. By default, community-level variance parameters for occupancy (`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 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 for each spatially-varying coefficient specified in `svc.cols`. 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 *` `length(svc.cols)` corresponding to the lower and upper support, respectively, or as a single value if the same value is assigned for all factor/SVC combinations. The priors for the factor loadings matrix `lambda` for each SVC are fixed following the standard spatial factor model to ensure parameter identifiability (Christensen and Amemlya 2002). The upper triangular elements of the N x `n.factors` matrix are fixed at 0 and the diagonal elements are fixed at 1 for each SVC. The lower triangular elements are assigned a standard normal prior (i.e., mean 0 and variance 1). `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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. `sigma.sq.t` and `rho` are the AR(1) variance and correlation parameters for the AR(1) zero-mean temporal random effects, respectively. `sigma.sq.t` is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which can each be specified as vector equal to the number of species in the model or a single value if the same prior is used for all species. `rho` is assumed to follow a uniform distribution, where the hyperparameters are specified similarly as a list of length two with the first and second elements corresponding to the lower and upper bounds of the uniform prior, which can each be specified as vector equal to the number of species in the model or a single value if the same prior is used for all species.

tuning   a list with each tag corresponding to a parameter name. Valid tags are `phi`, `nu`, and `rho`. The value portion of each tag defines the initial variance of the adaptive sampler. We assume the initial variance of the adaptive sampler is the same for each species, although the adaptive sampler will adjust the tuning variances separately for each species. 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 cor-

responding to variable names in occ.covs (for the intercept, use '(Intercept)').
svc.cols default argument of 1 results in a spatial occupancy model analogous
to sfMsPGOcc (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. 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 gener-
                 ally 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 pro-
                 duce 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.

std.by.sp   a logical value indicating whether the covariates are standardized separately
                 for each species within the corresponding range for each species (TRUE) or not
                 (FALSE). Note that if range.ind is specified in data.list, this will result in
                 the covariates being standardized differently for each species based on the sites
                 where range.ind == 1 for that given species. If range.ind is not specified and
                 std.by.sp = TRUE, this will simply be equivalent to standardizing the covariates
                 across all locations prior to fitting the model.

n.factors   the number of factors to use in the spatial factor model approach. Note this
                 corresponds to the number of factors used for each spatially-varying coefficient
                 that is estimated in the model. 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).

svc.by.sp   an optional list with length equal to length(svc.cols). Each element of the list
                 should be a logical vector of length N (number of species) where each element
                 is TRUE, which indicates the SVC should be estimated for that species, or 0,
                 which indicates the SVC should be set to 0 and no SVC for that parameter will be
                 estimated. Note the first n.factors SVCs for all spatially-varying coefficients
                 must be set to TRUE. By default, SVCs are modeled for all species.

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.
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.
ar1	logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If FALSE, the model is fit without an AR(1) temporal autocovariance structure. If TRUE, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.
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 svcTMsPGOcc that is a list comprised of:

beta.comm.samples

a coda object of posterior samples for the community level occurrence 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 occurrence 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 occurrence regression coefficients.
alpha.samples	a coda object of posterior samples for the species level detection regression coefficients.
theta.samples	a coda object of posterior samples for the species level correlation parameters for each spatially-varying coefficient and the temporal autocorrelation parameters for each species when ar1 = TRUE.

lambda.samples   a coda object of posterior samples for the latent spatial factor loadings for each spatially-varying coefficient.

z.samples   a four-dimensional array of posterior samples for the latent occurrence values for each species. Dimensions corresopnd to MCMC sample, species, site, and primary time period.

psi.samples   a four-dimensional array of posterior samples for the latent occupancy probability values for each species. Dimensions correspond to MCMC sample, species, site, and primary time period.

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.psi.samples

    a coda object of posterior samples for variances of random intercepts included in the occurrence portion of the model. Only included if random intercepts are specified in occ.formula.

sigma.sq.p.samples

    a coda object of posterior samples for variances of random intercpets included in the detection portion of the model. Only included if random intercepts are specified in det.formula.

beta.star.samples

    a coda object of posterior samples for the occurrence random effects. Only included if random intercepts are specified in occ.formula.

alpha.star.samples

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

like.samples   a four-dimensional array of posterior samples for the likelihood value used for calculating WAIC. Dimensions correspond to MCMC sample, species, site, and time period.

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().

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

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.

Finley, A. O., Datta, A., and Banerjee, S. (2020). spNNGP R package for nearest neighbor Gaussian process models. *arXiv* preprint arXiv:2001.09111.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. *Ecological Monographs*, 85(1), 3-28.

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

```
# Simulate Data ------------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
}
N <- 7
# Community-level covariate effects
# Occurrence
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
```

```
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- TRUE
svc.cols <- c(1, 2)
p.svc <- length(svc.cols)
n.factors <- 3
phi <- runif(p.svc * n.factors, 3 / .9, 3 / .3)
factor.model <- TRUE
cov.model <- 'exponential'

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
                 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
                 psi.RE = psi.RE, p.RE = p.RE, factor.model = factor.model,
                 svc.cols = svc.cols, n.factors = n.factors, phi = phi, sp = sp,
                 cov.model = cov.model)

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

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

data.list <- list(y = y, occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
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 / .9, b = 3 / .1))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                   alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
                   phi = 3 / .5, z = z.init)
# Tuning
tuning.list <- list(phi = 1)
```

```
# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

out <- svcTMsPGOcc(occ.formula = ~ occ.cov.1 + occ.cov.2,
                   det.formula = ~ det.cov.1 + det.cov.2,
                   data = data.list,
                   inits = inits.list,
                   n.batch = n.batch,
                   batch.length = batch.length,
                   accept.rate = 0.43,
                   NNGP = TRUE,
                   n.neighbors = 5,
                   n.factors = n.factors,
                   svc.cols = svc.cols,
                   cov.model = 'exponential',
                   priors = prior.list,
                   tuning = tuning.list,
                   n.omp.threads = 1,
                   verbose = TRUE,
                   n.report = 1,
                   n.burn = n.burn,
                   n.thin = n.thin,
                   n.chains = 1)

summary(out)
```

---

svcTPGBinom	*Function for Fitting Multi-Season Single-Species Spatially-Varying Coefficient Binomial Models Using Polya-Gamma Latent Variables*

Description

The function `svcTPGBinom` fits multi-season single-species spatially-varying coefficient binomial models using Polya-Gamma latent variables. Models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
svcTPGBinom(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, n.omp.threads = 1,
            verbose = TRUE, ar1 = FALSE, n.report = 100,
```

```
                n.burn = round(.10 * n.batch * batch.length),
                n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
                k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, covs, weights,
               and coords. y is a two-dimensional array with the rows corresponding to the
               number of sites (J) and columns corresponding to the maximum number of pri-
               mary time periods (i.e., years or seasons). covs is a list of variables included in
               the occurrence portion of the model. Each list element is a different occurrence
               covariate, which can be site level or site/primary time period level. Site-level
               covariates are specified as a vector of length J while site/primary time period
               level covariates are specified as a matrix with rows corresponding to sites and
               columns correspond to primary time periods. weights is a site by time period
               matrix containing the binomial weights (i.e., the total number of Bernoulli trials)
               at each site/time period combination. Note that missing values are allowed and
               should be specified as NA. coords is a $J \times 2$ matrix of the observation coordi-
               nates. Note that spOccupancy assumes coordinates are specified in a projected
               coordinate system.

inits   a list with each tag corresponding to a parameter name. Valid tags are beta,
               sigma.sq, phi, w, nu, sigma.sq.psi, sigma.sq.t, and rho. nu is only speci-
               fied if cov.model = "matern", and sigma.sq.psi is only specified if there are
               random effects in formula. sigma.sq.t and rho are only relevant when ar1 =
               TRUE. 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, sigma.sq.unif, nu.unif, sigma.sq.psi.ig, sigma.sq.t.ig,
               and rho.unif. Regression coefficients (beta) 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 2.73. The spatial variance parameter, sigma.sq,
               for each spatially-varying coefficient is assumed to follow an inverse-Gamma
               distribution or a uniform distribution (default is inverse-Gamma). The spatial
               decay phi and smoothness nu parameters are assumed to follow Uniform distri-
               butions. The hyperparameters of the inverse-Gamma for sigma.sq are passed
               as a list with two elements corresponding to the shape and scale parameters,
               respetively, with each element comprised of a vector equal to the number of
               spatially-varying coefficients to be estimated or of length one if priors are the
               same for all coefficients. The hyperparameters of any uniform priors are also

passed as a list of length two with the first and second elements corresponding to the lower and upper support, respectively, which can be passed as a vector equal to the total number of spatially-varying coefficients to be estimated or of length one if priors are the same for all coefficients. sigma.sq.psi are the random effect variances for any 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 or of length one if priors are the same for all random effect variances. sigma.sq.t and rho are the AR(1) variance and correlation parameters for the AR(1) zero-mean temporal random effects, respectively. sigma.sq.t is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a vector with elements corresponding to the shape and scale parameters, respectively. rho is assumed to follow a uniform distribution, where the hyperparameters are specified in a vector of length two with elements corresponding to the lower and upper bounds of the uniform prior.

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 covs (for the intercept, use '(Intercept)').

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. Valid tags are phi, sigma.sq, nu, and rho. The value portion of each tag defines the initial variance of the Adaptive sampler. See Roberts and Rosenthal (2009) for details.

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. Currently, only NNGP = TRUE is supported for multi-season occupancy models.

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 or k-fold cross-validation.

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.
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.
ar1	logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If FALSE, the model is fit without an AR(1) temporal autocovariance structure. If TRUE, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). For cross-validation in multi-season models, the data are split along the site dimension, such that each hold-out data set consists of a J / k.fold sites sampled over all primary time periods during which data are available at each given site. Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples   a coda object of posterior samples for the regression coefficients.

y.rep.samples   a three-dimensional array of posterior samples for the fitted data values, with
                   dimensions corresponding to posterior sample, site, and primary time period.

psi.samples   a three-dimensional array of posterior samples for the occurrence probability
                   values, with dimensions corresponding to posterior sample, site, and primary
                   time period.

theta.samples   a coda object of posterior samples for spatial covariance parameters and tempo-
                   ral covariance parameters if ar1 = TRUE.

w.samples   a three-dimensional array of posterior samples for the latent spatial random ef-
                   fects for all spatially-varying coefficients. Dimensions correspond to MCMC
                   sample, coefficient, and sites.

sigma.sq.psi.samples

                   a coda object of posterior samples for variances of unstructured random inter-
                   cepts included in the model. Only included if random intercepts are specified in
                   formula.

beta.star.samples

                   a coda object of posterior samples for the unstructured random effects. Only
                   included if random intercepts are specified in formula.

eta.samples   a coda object of posterior samples for the AR(1) random effects for each primary
                   time period. Only included if ar1 = TRUE.

like.samples   a three-dimensional array of posterior samples for the likelihood values associ-
                   ated with each site and primary time period. 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().

k.fold.deviance

                   soring rule (deviance) from k-fold cross-validation. Only included if k.fold is
                   specified in function call.

The return object will include additional objects used for subsequent prediction and/or model fit
evaluation. Note that if k.fold.only = TRUE, the return list object will only contain run.time and
k.fold.deviance

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.

Finley, A. O., and Banerjee, S. (2020). Bayesian spatially varying coefficient models in the spBayes R package. *Environmental Modelling and Software*, 125, 104608.

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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 <- 15
J.y <- 15
J <- J.x * J.y
# Years sampled
n.time <- sample(10, J, replace = TRUE)
# Binomial weights
weights <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  weights[j, 1:n.time[j]] <- sample(5, n.time[j], replace = TRUE)
}
# Occurrence ------------------------
beta <- c(-2, -0.5, -0.2, 0.75)
p.occ <- length(beta)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Spatial parameters -----------------
sp <- TRUE
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.1, 1)
phi <- runif(p.svc, 3/1, 3/0.2)
# Temporal parameters ----------------
ar1 <- TRUE
rho <- 0.8
sigma.sq.t <- 1

# Get all the data
dat <- simTBinom(J.x = J.x, J.y = J.y, n.time = n.time, weights = weights, beta = beta,
                 psi.RE = psi.RE, sp.only = sp.only, trend = trend,
                 sp = sp, svc.cols = svc.cols,
                 cov.model = cov.model, sigma.sq = sigma.sq, phi = phi,
```

```
                           rho = rho, sigma.sq.t = sigma.sq.t, ar1 = TRUE, x.positive = FALSE)

  # Prep the data for spOccupancy -----------------------------------------
  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],
               trend = X[, , 2],
               cov.1 = X[, , 3],
               cov.2 = X[, , 4])
  # Data list bundle
  data.list <- list(y = y,
                    covs = covs,
                    weights = weights,
                    coords = coords)
  # Priors
  prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                     sigma.sq.ig = list(a = 2, b = 1),
                     phi.unif = list(a = 3/1, b = 3/.1),
                     sigma.sq.t.ig = c(2, 0.5),
                     rho.unif = c(-1, 1))

  # Starting values
  inits.list <- list(beta = beta, alpha = 0,
                     sigma.sq = 1, phi = 3 / 0.5,
                     sigma.sq.t = 0.5, rho = 0)
  # Tuning
  tuning.list <- list(phi = 0.4, nu = 0.3, rho = 0.2)

  # MCMC settings
  n.batch <- 2
  n.burn <- 0
  n.thin <- 1

  out <- svcTPGBinom(formula = ~ trend + cov.1 + cov.2,
                     svc.cols = svc.cols,
                     data = data.list,
                     n.batch = n.batch,
                     batch.length = 25,
                     inits = inits.list,
                     priors = prior.list,
                     accept.rate = 0.43,
                     cov.model = "exponential",
                     ar1 = TRUE,
                     tuning = tuning.list,
                     n.omp.threads = 1,
                     verbose = TRUE,
                     NNGP = TRUE,
                     n.neighbors = 5,
                     n.report = 1,
```

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

---

svcTPGOcc   *Function for Fitting Multi-Season Single-Species Spatially-Varying*
                                Coefficient Occupancy Models Using Polya-Gamma Latent Variables

---

Description

Function for fitting multi-season single-species spatially-varying coefficient occupancy models using Polya-Gamma latent variables. Models are fit using Nearest Neighbor Gaussian Processes.

Usage

```
svcTPGOcc(occ.formula, det.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, n.omp.threads = 1,
          verbose = TRUE, ar1 = FALSE, n.report = 100,
          n.burn = round(.10 * n.batch * batch.length),
          n.thin = 1, n.chains = 1, k.fold, k.fold.threads = 1,
          k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
                det.covs, and coords. y is a three-dimensional array with first dimension equal
                to the number of sites (J), second dimension equal to the maximum number of
                primary time periods (i.e., years or seasons), and third dimension equal to the
                maximum number of replicates at a given site. occ.covs is a list of variables
                included in the occurrence portion of the model. Each list element is a different
                occurrence covariate, which can be site level or site/primary time period level.
                Site-level covariates are specified as a vector of length J while site/primary time
                period level covariates are specified as a matrix with rows corresponding to sites
                and columns correspond to primary time periods. Similarly, det.covs is a list
                of variables included in the detection portion of the model, with each list ele-
                ment corresponding to an individual variable. In addition to site-level and/or

site/primary time period-level, detection covariates can also be observational-level. Observation-level covariates are specified as a three-dimensional array with first dimension corresponding to sites, second dimension corresponding to primary time period, and third dimension corresponding to replicate. coords is a $J \times 2$ matrix of the observation coordinates. Note that spOccupancy assumes coordinates are specified in a projected coordinate system.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta, alpha, sigma.sq, phi, w, nu, sigma.sq.psi, sigma.sq.p, sigma.sq.t, rho. The value portion of each tag is the parameter's initial value. sigma.sq.psi and sigma.sq.p are only relevant when including random effects in the occurrence and detection portion of the occupancy model, respectively. nu is only specified if cov.model = "matern". sigma.sq.t and rho are only relevant when ar1 = TRUE. 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, sigma.sq.psi.ig, sigma.sq.p.ig, phi.unif, sigma.sq.ig, nu.unif, sigma.sq.t.ig, and rho.unif. Occupancy (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 2.72. sigma.sq.psi and sigma.sq.p are the random effect variances for any occurrence 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. The spatial variance parameter, sigma.sq, is assumed to follow an inverse-Gamma distribution. The spatial decay phi and smoothness nu parameters are assumed to follow Uniform distributions. The hyperparameters of the inverse-Gamma for sigma.sq.ig are passed as a list of length two, with the first and second elements corresponding to the shape and scale parameters, respectively, with each element comprised of a vector equal to the number of spatially-varying coefficients to be estimated or of length one if priors are the same for all coefficients. 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, which can be passed as a vector equal to the number of spatially-varying coefficients to be estimated or of length one if priors are the same for all coefficients. sigma.sq.t and rho are the AR(1) variance and correlation parameters for the AR(1) zero-mean temporal random effects, respectively. sigma.sq.t is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a vector with elements corresponding to the shape and scale parameters, respectively. rho is assumed to follow a uniform distribution, where the hyperparam-

eters are specified in a vector of length two with elements corresponding to the lower and upper bounds of the uniform prior.

tuning   a list with each tag corresponding to a parameter name. Valid tags are phi, sigma.sq, nu, and rho. The value portion of each tag defines the initial variance of the Adaptive sampler. 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 occupancy model analogous to stPGOcc (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. Currently only NNGP = TRUE is supported for multi-season single-species occupancy models.

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 or k-fold cross-validation.

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.

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. 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.

ar1	logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If FALSE, the model is fit without an AR(1) temporal autocovariance structure. If TRUE, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). For cross-validation in multi-season models, the data are split along the site dimension, such that each hold-out data set consists of a J / k.fold sites sampled over all primary time periods during which data are available at each given site. Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.
k.fold.only	a logical value indicating whether to only perform cross-validation (TRUE) or perform cross-validation after fitting the full model (FALSE). Default value is FALSE.
...	currently no additional arguments

Value

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

beta.samples	a coda object of posterior samples for the occupancy regression coefficients.
alpha.samples	a coda object of posterior samples for the detection regression coefficients.
z.samples	a three-dimensional array of posterior samples for the latent occupancy values, with dimensions corresponding to posterior sample, site, and primary time period.
psi.samples	a three-dimensional array of posterior samples for the latent occupancy probability values, with dimensions corresponding to posterior sample, site, and primary time period.

theta.samples   a coda object of posterior samples for spatial covariance parameters and tempo-
                 ral covariance parameters if ar1 = TRUE.

w.samples   a three-dimensional array of posterior samples for the latent spatial random ef-
                 fects for all spatially-varying coefficients. Dimensions correspond to MCMC
                 sample, coefficient, and sites.

sigma.sq.psi.samples
                 a coda object of posterior samples for variances of random intercepts included
                 in the occupancy portion of the model. Only included if random intercepts are
                 specified in occ.formula.

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

beta.star.samples
                 a coda object of posterior samples for the occurrence random effects. Only
                 included if random intercepts are specified in occ.formula.

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

eta.samples   a coda object of posterior samples for the AR(1) random effects for each primary
                 time period. Only included if ar1 = TRUE.

like.samples   a three-dimensional array of posterior samples for the likelihood values associ-
                 ated with each site and primary time period. 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().

k.fold.deviance
                 scoring rule (deviance) from k-fold cross-validation. Only included if k.fold is
                 specified in function call.

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(). Note that if k.fold.only = TRUE, the return list object will
only contain run.time and k.fold.deviance.

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribu-
tion is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code
implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.
lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Kery, M., & Royle, J. A. (2021). Applied hierarchical modeling in ecology: Analysis of distribution, abundance and species richness in R and BUGS: Volume 2: Dynamic and advanced models. Academic Press. Section 4.6.

Finley, A. O., and Banerjee, S. (2020). Bayesian spatially varying coefficient models in the spBayes R package. *Environmental Modelling and Software*, 125, 104608.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm. 2002. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology 83: 2248-2255.

Examples

```
set.seed(1000)
# Sites
J.x <- 15
J.y <- 15
J <- J.x * J.y
# Years sampled
n.time <- sample(10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(4, n.time[j], replace = TRUE)
}
# Occurrence -------------------------
beta <- c(-2, -0.5, -0.2, 0.75)
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection --------------------------
alpha <- c(1, 0.7, -0.5)
p.RE <- list()
# Spatial parameters -----------------
sp <- TRUE
svc.cols <- c(1, 2, 3)
p.svc <- length(svc.cols)
cov.model <- "exponential"
sigma.sq <- runif(p.svc, 0.1, 1)
phi <- runif(p.svc, 3 / 1, 3 / 0.2)
rho <- 0.8
sigma.sq.t <- 1
ar1 <- TRUE
```

```
x.positive <- FALSE

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
               beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
               psi.RE = psi.RE, p.RE = p.RE,
               sp = sp, cov.model = cov.model, sigma.sq = sigma.sq, phi = phi,
               svc.cols = svc.cols, ar1 = ar1, rho = rho, sigma.sq.t = sigma.sq.t,
               x.positive = x.positive)

# Prep the data for svcTPGOcc --------------------------------------------
# Full data set
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

# Package all data into a list
occ.covs <- list(int = X[, , 1],
                 trend = X[, , 2],
                 occ.cov.1 = X[, , 3],
                 occ.cov.2 = X[, , 4])
# Detection
det.covs <- list(det.cov.1 = X.p[, , , 2],
                 det.cov.2 = X.p[, , , 3])
# Data list bundle
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  coords = coords)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   phi.unif = list(a = 3/1, b = 3/.1))

# Starting values
z.init <- apply(y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0,
                   sigma.sq = 1, phi = 3 / 0.5,
                   z = z.init, nu = 1)
# Tuning
tuning.list <- list(phi = 0.4, nu = 0.3, rho = 0.5, sigma.sq = 0.5)

# MCMC settings
n.batch <- 2
n.burn <- 0
n.thin <- 1

# Run the model
out <- svcTPGOcc(occ.formula = ~ trend + occ.cov.1 + occ.cov.2,
                 det.formula = ~ det.cov.1 + det.cov.2,
```

```
              data = data.list,
              inits = inits.list,
              tuning = tuning.list,
              priors = prior.list,
              cov.model = "exponential",
              svc.cols = svc.cols,
              NNGP = TRUE,
              ar1 = TRUE,
              n.neighbors = 5,
              n.batch = n.batch,
              batch.length = 25,
              verbose = TRUE,
              n.report = 25,
              n.burn = n.burn,
              n.thin = n.thin,
              n.chains = 1)
```

---

tMsPGOcc   *Function for Fitting Multi-Species Multi-Season Occupancy Models*

Description

The function `tMsPGOcc` fits multi-species multi-season occupancy models using Polya-Gamma data
augmentation.

Usage

```
tMsPGOcc(occ.formula, det.formula, data, inits, priors, tuning,
         n.batch, batch.length,
         accept.rate = 0.43, n.omp.threads = 1,
         verbose = TRUE, ar1 = FALSE, n.report = 100,
         n.burn = round(.10 * n.batch * batch.length), n.thin = 1,
         n.chains = 1, ...)
```

Arguments

occ.formula	a symbolic description of the model to be fit for the occurrence portion of the model using R's model syntax. Random intercepts are allowed using lme4 syntax (Bates et al. 2015). Only right-hand side of formula is specified. See example below.
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 are allowed using lme4 syntax (Bates et al. 2015).
data	a list containing data necessary for model fitting. Valid tags are y, occ.covs, and det.covs. y is a four-dimensional array with first dimension equal to the number of species, second dimension equal to the number of sites, third dimension equal to the number of primary time periods, and fourth dimension equal

to the maximum number of secondary replicates at a given site. `occ.covs` is a list of variables included in the occurrence portion of the model. Each list element is a different occurrence covariate, which can be site level or site/primary time period level. Site-level covariates are specified as a vector of length J while site/primary time period level covariates are specified as a matrix with rows corresponding to sites and columns correspond to primary time periods. Similarly, `det.covs` is a list of variables included in the detection portion of the model, with each list element corresponding to an individual variable. In addition to site-level and/or site/primary time period-level, detection covariates can also be observational-level. Observation-level covariates are specified as a three-dimensional array with first dimension corresponding to sites, second dimension corresponding to primary time period, and third dimension corresponding to replicate.

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.psi`, `sigma.sq.p`, `z`, `sigma.sq.t`, and `rho`. `sigma.sq.t` and `rho` are only relevant when `ar1 = TRUE`, and `sigma.sq.psi` and `sigma.sq.p` are only specified if random effects are included in `occ.formula` or `det.formula`, respectively. 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.psi`, `sigma.sq.p`, `sigma.sq.t.ig`, and `rho.unif`. Community-level occurrence (`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 set to 2.72. By default, community-level variance parameters for occupancy (`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 priors are the same for all parameters. If not specified, prior shape and scale parameters are set to 0.1. `sigma.sq.t` and `rho` are the AR(1) variance and correlation parameters for the AR(1) zero-mean temporal random effects, respectively. `sigma.sq.t` is assumed to follow an inverse-Gamma distribution, where the hyperparameters are specified as a list of length two with the first and second elements corresponding to the shape and scale parameters, respectively, which can each be specified as vector equal to the number of species in the model or a single value if the same prior is used for all species. `rho` is assumed to follow a uniform distribution, where the hyperparameters are specified similarly as a list of length two with the first and second elements corresponding to the lower and upper bounds of the uniform prior, which can each be specified as

vector equal to the number of species in the model or a single value if the same prior is used for all species. `sigma.sq.psi` and `sigma.sq.p` are the random effect variances for any occurrence 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.

tuning
: a list with each tag corresponding to a parameter name. Valid tags are rho. The value portion of each tag defines the initial tuning variance of the Adaptive sampler. 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 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.

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.

ar1
: logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model for each species. If FALSE, the model is fit without an AR(1) temporal autocovariance structure. If TRUE, a species-specific AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.

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 `tMsPGOcc` that is a list comprised of:

beta.comm.samples
: a coda object of posterior samples for the community level occurrence 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 occurrence community variance pa-
                   rameters.

tau.sq.alpha.samples
                   a coda object of posterior samples for the detection community variance param-
                   eters.

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

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

theta.samples   a coda object of posterior samples for the species level AR(1) variance (sigma.sq.t)
                   and correlation (rho) parameters. Only included if ar1 = TRUE.

eta.samples   a three-dimensional array of posterior samples for the species-specific AR(1)
                   random effects for each primary time period. Dimensions correspond to MCMC
                   sample, species, and primary time period.

z.samples   a four-dimensional array of posterior samples for the latent occurrence values
                   for each species. Dimensions corresopnd to MCMC sample, species, site, and
                   primary time period.

psi.samples   a four-dimensional array of posterior samples for the latent occupancy probabil-
                   ity values for each species. Dimensions correspond to MCMC sample, species,
                   site, and primary time period.

sigma.sq.psi.samples
                   a coda object of posterior samples for variances of random intercepts included
                   in the occurrence portion of the model. Only included if random intercepts are
                   specified in occ.formula.

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

beta.star.samples
                   a coda object of posterior samples for the occurrence random effects. Only
                   included if random intercepts are specified in occ.formula.

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

like.samples   a four-dimensional array of posterior samples for the likelihood value used for
                   calculating WAIC. Dimensions correspond to MCMC sample, species, site, and
                   time period.

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()`.

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Kery, M., & Royle, J. A. (2021). Applied hierarchical modeling in ecology: Analysis of distribution, abundance and species richness in R and BUGS: Volume 2: Dynamic and advanced models. Academic Press. Section 4.6.

Examples

```
# Simulate Data ------------------------------------------------------------
set.seed(500)
J.x <- 8
J.y <- 8
J <- J.x * J.y
# Years sampled
n.time <- sample(3:10, J, replace = TRUE)
# n.time <- rep(10, J)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(2:4, n.time[j], replace = TRUE)
}
N <- 7
# Community-level covariate effects
# Occurrence
beta.mean <- c(-3, -0.2, 0.5)
trend <- FALSE
sp.only <- 0
```

```
p.occ <- length(beta.mean)
tau.sq.beta <- c(0.6, 1.5, 1.4)
# Detection
alpha.mean <- c(0, 1.2, -1.5)
tau.sq.alpha <- c(1, 0.5, 2.3)
p.det <- length(alpha.mean)
# Random effects
psi.RE <- list()
p.RE <- list()
# Draw species-level effects from community means.
beta <- matrix(NA, nrow = N, ncol = p.occ)
alpha <- matrix(NA, nrow = N, ncol = p.det)
for (i in 1:p.occ) {
  beta[, i] <- rnorm(N, beta.mean[i], sqrt(tau.sq.beta[i]))
}
for (i in 1:p.det) {
  alpha[, i] <- rnorm(N, alpha.mean[i], sqrt(tau.sq.alpha[i]))
}
sp <- FALSE

dat <- simTMsOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep, N = N,
                 beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
                 psi.RE = psi.RE, p.RE = p.RE, sp = sp)

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

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

data.list <- list(y = y, occ.covs = occ.covs,
                   det.covs = det.covs)
# Priors
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))
z.init <- apply(y, c(1, 2, 3), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(alpha.comm = 0, beta.comm = 0, beta = 0,
                    alpha = 0, tau.sq.beta = 1, tau.sq.alpha = 1,
                    z = z.init)
# Tuning
tuning.list <- list(phi = 1)

# Number of batches
n.batch <- 5
# Batch length
batch.length <- 25
```

```
n.burn <- 25
n.thin <- 1
n.samples <- n.batch * batch.length

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

summary(out)
```

---

tPGOcc	*Function for Fitting Multi-Season Single-Species Occupancy Models Using Polya-Gamma Latent Variables*

Description

Function for fitting multi-season single-species occupancy models using Polya-Gamma latent variables.

Usage

```
tPGOcc(occ.formula, det.formula, data, inits, priors, tuning,
       n.batch, batch.length, accept.rate = 0.43, n.omp.threads = 1,
       verbose = TRUE, ar1 = FALSE, n.report = 100,
       n.burn = round(.10 * n.batch * batch.length), n.thin = 1, n.chains = 1,
       k.fold, k.fold.threads = 1, k.fold.seed = 100, k.fold.only = FALSE, ...)
```

Arguments

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

data   a list containing data necessary for model fitting. Valid tags are y, occ.covs,
                and det.covs. y is a three-dimensional array with first dimension equal to
                the number of sites (J), second dimension equal to the maximum number of
                primary time periods (i.e., years or seasons), and third dimension equal to the
                maximum number of replicates at a given site. occ.covs is a list of variables
                included in the occurrence portion of the model. Each list element is a different
                occurrence covariate, which can be site level or site/primary time period level.
                Site-level covariates are specified as a vector of length J while site/primary time
                period level covariates are specified as a matrix with rows corresponding to sites
                and columns correspond to primary time periods. Similarly, det.covs is a list
                of variables included in the detection portion of the model, with each list ele-
                ment corresponding to an individual variable. In addition to site-level and/or
                site/primary time period-level, detection covariates can also be observational-
                level. Observation-level covariates are specified as a three-dimensional array
                with first dimension corresponding to sites, second dimension corresponding to
                primary time period, and third dimension corresponding to replicate.

inits   a list with each tag corresponding to a parameter name. Valid tags are z, beta,
                alpha, sigma.sq.psi, sigma.sq.p, sigma.sq.t, and rho. The value portion
                of each tag is the parameter's initial value. sigma.sq.psi and sigma.sq.p are
                only relevant when including random effects in the occurrence and detection
                portion of the occupancy model, respectively. sigma.sq.t and rho are only
                relevant when ar1 = TRUE. See priors description for definition of each param-
                eter 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, sigma.sq.psi.ig, sigma.sq.p.ig, sigma.sq.t.ig, and rho.unif.
                Occupancy (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 2.72. sigma.sq.psi and sigma.sq.p are the
                random effect variances for any unstructured occurrence 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. sigma.sq.t and rho are the AR(1) variance and
                correlation parameters for the AR(1) zero-mean temporal random effects, re-
                spectively. sigma.sq.t is assumed to follow an inverse-Gamma distribution,
                where the hyperparameters are specified as a vector with elements correspond-
                ing to the shape and scale parameters, respectively. rho is assumed to follow
                a uniform distribution, where the hyperparameters are specified in a vector of
                length two with elements corresponding to the lower and upper bounds of the
                uniform prior.

tuning	a list with each tag corresponding to a parameter name. Valid tags are rho. The value portion of each tag defines the initial tuning variance of the Adaptive sampler. 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.
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. 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.
ar1	logical value indicating whether to include an AR(1) zero-mean temporal random effect in the model. If FALSE, the model is fit without an AR(1) temporal autocovariance structure. If TRUE, an AR(1) random effect is included in the model to account for temporal autocorrelation across the primary time periods.
n.report	the interval to report MCMC progress. Note this is specified in terms of batches, not MCMC samples.
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.
k.fold	specifies the number of *k* folds for cross-validation. If not specified as an argument, then cross-validation is not performed and k.fold.threads and k.fold.seed are ignored. In *k*-fold cross-validation, the data specified in data is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, *k* - 1 subsamples are used to fit the model and the remaining *k* samples are used for prediction. The cross-validation process is repeated *k* times (the folds). As a scoring rule, we use the model deviance as described in Hooten and Hobbs (2015). For cross-validation in multi-season models, the data are split along the site dimension, such that each hold-out data set consists of J / k.fold sites sampled over all primary time periods during which data are available at each given site. Cross-validation is performed after the full model is fit using all the data. Cross-validation results are reported in the k.fold.deviance object in the return list.
k.fold.threads	number of threads to use for cross-validation. If k.fold.threads > 1 parallel processing is accomplished using the **foreach** and **doParallel** packages. Ignored if k.fold is not specified.
k.fold.seed	seed used to split data set into k.fold parts for k-fold cross-validation. Ignored if k.fold is not specified.

k.fold.only   a logical value indicating whether to only perform cross-validation (TRUE) or
                   perform cross-validation after fitting the full model (FALSE). Default value is
                   FALSE.

...   currently no additional arguments

Value

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

beta.samples   a coda object of posterior samples for the occupancy regression coefficients.

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

z.samples   a three-dimensional array of posterior samples for the latent occupancy values,
                   with dimensions corresponding to posterior sample, site, and primary time pe-
                   riod. Note this object will contain predicted occupancy values for sites/primary
                   time periods that were not sampled.

psi.samples   a three-dimensional array of posterior samples for the latent occupancy probabil-
                   ity values, with dimensions corresponding to posterior sample, site, and primary
                   time period. Note this object will contain predicted occupancy probabilities for
                   sites/primary time periods that were not sampled.

sigma.sq.psi.samples
                   a coda object of posterior samples for variances of random intercepts included
                   in the occupancy portion of the model. Only included if random intercepts are
                   specified in occ.formula.

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

beta.star.samples
                   a coda object of posterior samples for the occurrence random effects. Only
                   included if random intercepts are specified in occ.formula.

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

theta.samples   a coda object of posterior samples for the AR(1) variance (sigma.sq.t) and
                   correlation (rho) parameters. Only included if ar1 = TRUE.

eta.samples   a coda object of posterior samples for the AR(1) random effects for each primary
                   time period. Only included if ar1 = TRUE.

like.samples   a three-dimensional array of posterior samples for the likelihood values associ-
                   ated with each site and primary time period. 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().

k.fold.deviance
                   scoring rule (deviance) from k-fold cross-validation. Only included if k.fold is
                   specified in function call.

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()`. Note that if `k.fold.only = TRUE`, the return list object will only contain `run.time` and `k.fold.deviance`.

Note

Some of the underlying code used for generating random numbers from the Polya-Gamma distribution is taken from the **pgdraw** package written by Daniel F. Schmidt and Enes Makalic. Their code implements Algorithm 6 in PhD thesis of Jesse Bennett Windle (2013) https://repositories.lib.utexas.edu/handle/2152/21842.

Author(s)

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

References

Polson, N.G., J.G. Scott, and J. Windle. (2013) Bayesian Inference for Logistic Models Using Polya-Gamma Latent Variables. *Journal of the American Statistical Association*, 108:1339-1349.

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.

Kery, M., & Royle, J. A. (2021). Applied hierarchical modeling in ecology: Analysis of distribution, abundance and species richness in R and BUGS: Volume 2: Dynamic and advanced models. Academic Press. Section 4.6.

Hooten, M. B., and Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.

MacKenzie, D. I., J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm. 2002. Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One. Ecology 83: 2248-2255.

Examples

```
set.seed(500)
# Sites
J.x <- 10
J.y <- 10
J <- J.x * J.y
# Primary time periods
n.time <- sample(5:10, J, replace = TRUE)
n.time.max <- max(n.time)
# Replicates
n.rep <- matrix(NA, J, max(n.time))
for (j in 1:J) {
  n.rep[j, 1:n.time[j]] <- sample(1:4, n.time[j], replace = TRUE)
}
# Occurrence ------------------------
beta <- c(0.4, 0.5, -0.9)
```

```
trend <- TRUE
sp.only <- 0
psi.RE <- list()
# Detection -------------------------
alpha <- c(-1, 0.7, -0.5)
p.RE <- list()
# Temporal parameters ----------------
rho <- 0.7
sigma.sq.t <- 0.6

# Get all the data
dat <- simTOcc(J.x = J.x, J.y = J.y, n.time = n.time, n.rep = n.rep,
                beta = beta, alpha = alpha, sp.only = sp.only, trend = trend,
                psi.RE = psi.RE, p.RE = p.RE, sp = FALSE, ar1 = TRUE,
                sigma.sq.t = sigma.sq.t, rho = rho)

# Package all data into a list
# Occurrence
occ.covs <- list(int = dat$X[, , 1],
                 trend = dat$X[, , 2],
                 occ.cov.1 = dat$X[, , 3])
# Detection
det.covs <- list(det.cov.1 = dat$X.p[, , , 2],
                 det.cov.2 = dat$X.p[, , , 3])
# Data list bundle
data.list <- list(y = dat$y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)
# Priors
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   rho.unif = c(-1, 1),
                   sigma.sq.t.ig = c(2, 0.5))

# Starting values
z.init <- apply(dat$y, c(1, 2), function(a) as.numeric(sum(a, na.rm = TRUE) > 0))
inits.list <- list(beta = 0, alpha = 0, z = z.init)

# Tuning
tuning.list <- list(rho = 0.5)

n.batch <- 20
batch.length <- 25
n.samples <- n.batch * batch.length
n.burn <- 100
n.thin <- 1

# Run the model
out <- tPGOcc(occ.formula = ~ trend + occ.cov.1,
              det.formula = ~ det.cov.1 + det.cov.2,
              data = data.list,
              inits = inits.list,
              priors = prior.list,
```

```
                    tuning = tuning.list,
                    n.batch = n.batch,
                    batch.length = batch.length,
                    verbose = TRUE,
                    ar1 = TRUE,
                    n.report = 25,
                    n.burn = n.burn,
                    n.thin = n.thin,
                    n.chains = 1)

    summary(out)
```

---

waicOcc	*Compute Widely Applicable Information Criterion for spOccupancy Model Objects*

Description

Function for computing the Widely Applicable Information Criterion (WAIC; Watanabe 2010) for spOccupancy model objects.

Usage

```
waicOcc(object, by.sp = FALSE, ...)
```

Arguments

object	an object of class PGOcc, spPGOcc, msPGOcc, spMsPGOcc, intPGOcc, spIntPGOcc, lfJSDM, sfJSDM, lfMsPGOcc, sfMsPGOcc, tPGOcc, stPGOcc, svcPGBinom, svcPGOcc, svcTPGBinom, svcTPGOcc, or intMsPGOcc, svcMsPGOcc, tMsPGOcc, stMsPGOcc, svcTMsPGOcc.
by.sp	a logical value indicating whether to return a separate WAIC value for each species in a multi-species occupancy model or a single value for all species.
...	currently no additional arguments

Details

The effective number of parameters is calculated following the recommendations of Gelman et al. (2014).

Value

When object is of class PGOcc, spPGOcc, msPGOcc, spMsPGOcc, lfJSDM, sfJSDM, lfMsPGOcc, sfMsPGOcc, tPGOcc, stPGOcc, svcPGBinom, svcPGOcc, svcTPGOcc, svcTPGBinom, svcMsPGOcc, tMsPGOcc, stMsPGOcc, svcTMsPGOcc 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. When by.sp = TRUE for multi-species models, object is a data frame with each row corresponding to a different species. When object is of class intPGOcc or spIntPGOcc, returns a data frame with columns elpd, pD, and WAIC, with each row corresponding to the estimated values for each data source in the integrated model.

Author(s)

Jeffrey W. Doser <doserjef@msu.edu>,
Andrew O. Finley <finleya@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(400)
# Simulate Data -----------------------------------------------------------
J.x <- 8
J.y <- 8
J <- J.x * J.y
n.rep <- sample(2:4, J, replace = TRUE)
beta <- c(0.5, -0.15)
p.occ <- length(beta)
alpha <- c(0.7, 0.4)
p.det <- length(alpha)
dat <- simOcc(J.x = J.x, J.y = J.y, n.rep = n.rep, beta = beta, alpha = alpha,
              sp = FALSE)
occ.covs <- dat$X[, 2, drop = FALSE]
colnames(occ.covs) <- c('occ.cov')
det.covs <- list(det.cov = dat$X.p[, , 2])
# Data bundle
data.list <- list(y = dat$y,
                  occ.covs = occ.covs,
                  det.covs = det.covs)

# Priors
prior.list <- list(beta.normal = list(mean = rep(0, p.occ),
                                       var = rep(2.72, p.occ)),
                   alpha.normal = list(mean = rep(0, p.det),
                                       var = rep(2.72, p.det)))
# Initial values
inits.list <- list(alpha = rep(0, p.det),
                   beta = rep(0, p.occ),
                   z = apply(data.list$y, 1, max, na.rm = TRUE))

n.samples <- 5000
n.report <- 1000

out <- PGOcc(occ.formula = ~ occ.cov,
             det.formula = ~ det.cov,
```

```
              data = data.list,
              inits = inits.list,
              n.samples = n.samples,
              priors = prior.list,
              n.omp.threads = 1,
              verbose = TRUE,
              n.report = n.report,
              n.burn = 4000,
              n.thin = 1)

# Calculate WAIC
waicOcc(out)
```

Index