Package ‘MultiNMix’

December 26, 2024

Title Multi-Species N-Mixture (MNM) Models with Nimble

Version 0.1.0

Description Tools for simulating data and fitting multi-species N-mixture (MNM) models using Nimble. Includes features for handling zero-inflation and temporal correlation, Bayesian inference, model diagnostics, parameter estimation, and predictive checks. Designed for ecological studies with zero-altered or time-series data.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports clusterGeneration, mvtnorm, extraDistr, rstan, stats, abind, methods, coda

Depends nimble, R (>= 3.5)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

LazyData true

URL `https://github.com/niamhmimnagh/MNM`

BugReports `https://github.com/niamhmimnagh/MNM/blob/main/.github/ISSUE_TEMPLATE/bug_report.md`

NeedsCompilation no

Author Niamh Mimnagh [aut, cre],
Rafael de Andrade Moral [aut]

Maintainer Niamh Mimnagh <`niamhmimnagh@gmail.com`>

Contents

`AIC,MNM-method` *AIC Method for "MNM" Class*

Description

Computes the Akaike Information Criterion (AIC) for an object of class `"MNM"`. AIC is a metric used for model comparison, balancing goodness of fit and model complexity.

The formula for AIC is:

$$AIC = -2 \cdot \log L + 2 \cdot k$$

where:

- $\log L$ is the log-likelihood of the model.
- k is the number of parameters in the model.

Usage

```
## S4 method for signature 'MNM'
AIC(object)
```

Arguments

`object`	An object of class `"MNM"`.

Value

A numeric value representing the AIC of the fitted model.

`BIC,MNM-method` *BIC Method for "MNM" Class*

Description

Computes the Bayesian Information Criterion (BIC) for an object of class `"MNM"`. BIC is a metric used for model comparison, accounting for goodness of fit and the size of the dataset.

The formula for BIC is:

$$BIC = -2 \cdot \log L + \log(n) \cdot k$$

where:

- $\log L$ is the log-likelihood of the model.
- n is the number of observations in the dataset.
- k is the number of parameters in the model.

Usage

```
## S4 method for signature 'MNM'
BIC(object)
```

Arguments

`object`	An object of class `"MNM"`.

Value

A numeric value representing the BIC of the fitted model.

`birds` *Birds Dataset - Processed Subset of the North American Breeding Bird Survey Dataset*

Description

#' This dataset is a subset of the North American Breeding Bird Survey, containing data collected at 24 routes in Michigan, USA. Each route has 10 stops, and the dataset includes counts for 20 bird species. It is in the form of an array of dimension (R,T,S,K), which means that it is ready for use with the MNM_fit function and requires no further pre-processing.

Usage

```
birds
```

Format

An array of dimensions(24, 10, 20, 6), generated by processing the birds_raw dataset as in Example 1.

Source

North American Breeding Bird Survey (`https://www.pwrc.usgs.gov/BBS/`)

`birds_raw` *Raw Birds Dataset - Subset of the North American Breeding Bird Survey Dataset*

Description

This dataset is a subset of the North American Breeding Bird Survey, containing data collected at 24 routes in Michigan, USA. Each route has 10 stops, and the dataset includes counts for 20 bird species.

Usage

```
birds_raw
```

Format

A data frame with 2,880 rows and 53 columns:

Route An identifier for the 24 survey sites.

Year The year of the survey (numeric).

English_Common_Name The English common name of the bird species surveyed (character).

Stop1 Count for replicate 1 at the site (numeric).

Stop2 Count for replicate 2 at the site (numeric).

Stop3 Count for replicate 3 at the site (numeric).

Stop4 Count for replicate 4 at the site (numeric).

Stop5 Count for replicate 5 at the site (numeric).

Stop6 Count for replicate 6 at the site (numeric).

Stop7 Count for replicate 7 at the site (numeric).

Stop8 Count for replicate 8 at the site (numeric).

Stop9 Count for replicate 9 at the site (numeric).

Stop10 Count for replicate 10 at the site (numeric).

Details

This dataset represents a subset of the North American Breeding Bird Survey. Data was collected in Michigan between 2005 and 2010, with observations for 20 bird species recorded at 24 routes, each surveyed 10 times. The dataset is used to study avian biodiversity and population trends.

Source

North American Breeding Bird Survey (`https://www.pwrc.usgs.gov/BBS/`)

Examples

```
data(birds_raw)
head(birds_raw)

# Example: MNM
# Data must first be reformatted to an array of dimension (R,T,S,K)
R <- 24
T <- 10
S <- 20
K <- 6
# Ensure data is ordered consistently
birds <- birds_raw[order(birds_raw$Route, birds_raw$Year, birds_raw$English_Common_Name), ]

# Create a 4D array with proper dimension
Y <- array(NA, dim = c(R, T, S, K))

# Map route, species, and year to indices
route_idx <- as.numeric(factor(birds_raw$Route))
species_idx <- as.numeric(factor(birds_raw$English_Common_Name))
year_idx <- as.numeric(factor(birds_raw$Year))

# Populate the array
stop_data <- as.matrix(birds_raw[, grep("^Stop", colnames(birds))])

for (i in seq_len(nrow(birds))) {
  Y[route_idx[i], , species_idx[i], year_idx[i]] <- stop_data[i, ]
  }

  # Assign dimnames
  dimnames(Y) <- list(
    Route = sort(unique(birds_raw$Route)),
      Stop = paste0("Stop", 1:T),
        Species = sort(unique(birds_raw$English_Common_Name)),
          Year = sort(unique(birds_raw$Year))
          )

# Selecting only 5 bird species and 1 year for analysis:
Y<-Y[,,1:5,1]

## Not run: model<-MNM_fit(Y=Y, AR=FALSE, Hurdle=FALSE)
```

`check_convergence` *Convergence Check Generic Function*

Description

A generic function for checking the convergence of an "MNM" object.

Usage

```
check_convergence(object)
```

Arguments

`object`	An object of class "MNM".

check_convergence,MNM-method — *Convergence Check for "MNM" Class*

Description

Checks the convergence status of an object of class "MNM" based on convergence diagnostics (e.g., Rhat values). Returns the number of parameters that meet the convergence criterion (Rhat < 1.1).

Usage

```
## S4 method for signature 'MNM'
check_convergence(object)
```

Arguments

`object`	An object of class "MNM".

Value

A numeric value indicating the number of converged parameters.

clean_envir — *Clean Environment of Temporary Objects*

Description

This function removes temporary objects from a specified environment. It is designed for use in contexts where intermediate objects are created and should be cleaned up after execution to prevent clutter or unintended behavior.

Usage

```
clean_envir(env_rm = .GlobalEnv)
```

Arguments

`env_rm`	An environment from which to remove objects. Defaults to the global environment (`.GlobalEnv`). This may be set to a different environment for more controlled cleanup.

Details

This is an auxiliary function that removes temporary objects created by Nimble from the environment.

Value

The function does not return any values.

`density,MNM-method` *Density Plot Method for "MNM" Class*

Description

Generates the density plot for a specified parameter stored in the MNM object.

Usage

```
## S4 method for signature 'MNM'
density(x, param, ...)
```

Arguments

`x`	An object of class `"MNM"`.
`param`	The name of the parameter to plot (e.g., `"N[8, 1]"`).
`...`	Additional arguments (not used).

Value

The density plot for the specified parameter is displayed.

Examples

```
# Calling the density function
## Not run: density(y, "N[10, 1]")
```

`distribution_args` *List of Valid Nimble Distributions and Their Arguments*

Description

A named list where each key is a valid Nimble distribution, and the value is a vector of the required arguments.

Usage

```
distribution_args
```

Format

A named list of length 19.

Details

This object is used internally to validate user-specified prior distributions and their arguments in Nimble models.

`logLik,MNM-method` *Log-Likelihood Method for "MNM" Class*

Description

Retrieves the log-likelihood value from an object of class "MNM".

Usage

```
## S4 method for signature 'MNM'
logLik(object)
```

Arguments

`object`	An object of class "MNM".

Value

A numeric value representing the log-likelihood of the fitted model.

`MNM` *Fit a Multi-Species N-Mixture Model (MNM) using Nimble*

Description

Fits a multi-species N-mixture (MNM) model to observed count data, leveraging Nimble for Bayesian inference. This model accounts for variation in detection probability and abundance across multiple species and sites.

Usage

```
MNM(
  Y = NULL,
  iterations = 60000,
  burnin = 20000,
  thin = 10,
  Xp = NULL,
  Xn = NULL,
  ...
)
```

Arguments

`Y`	An array of observed count data of dimension (R, T, S), where: • `R`: Number of sites • `T`: Number of replicates • `S`: Number of species. This array is typically produced by the `simulateData` function.
`iterations`	Integer. Total number of MCMC iterations to run. Default is 60,000.

`burnin`	Integer. Number of initial MCMC iterations to discard as burn-in. Default is 20,000.
`thin`	Integer. Thinning interval for MCMC samples to reduce autocorrelation. Default is 10.
`Xp`	An array of covariates affecting detection probability, with dimensions (R, S, P1), where: • `R`: Number of sites • `S`: Number of species • `P1`: Number of detection-related covariates. See examples for implementation details.
`Xn`	An array of covariates affecting abundance, with dimensions (R, S, P2), where: • `R`: Number of sites • `S`: Number of species • `P1`: Number of abundance-related covariates. See examples for implementation details.
`...`	Additional arguments passed for prior distribution specification. Supported distributions include dnorm, dexp, dgamma, dbeta, dunif, dlnorm, dbern, dpois, dbinom, dcat, dmnorm, dwish, dchisq, dinvgamma, dt, dweib, ddirch, dmulti, dmvt. Default prior distributions are: • prior_detection_probability: prior distribution for the detection probability intercept (`gamma`). Default is `'dnorm(0, 0.001)'`. • prior_precision: prior distribution for the precision matrix for the species-level random effect. Default is `'dwish(Omega[1:S,1:S], df)'`. • prior_mean: prior distribution for the mean of the species-level random effect (`mu`). Default is `'dnorm(0,0.001)'`. • prior_hurdle: prior distribution for `theta`, the probability of structural zero in hurdle models. Default is `'dbeta(1,1)'`. • prior_mean_AR: prior distribution for the mean of the autoregressive random effect (`phi`). Default is `'dnorm(0,0.001)'`. • prior_sd_AR: prior distribution for the standard deviation of the autoregressive random effect (`phi`). Default is `'dexp(1)'`. See Nimble (r-nimble.org) documentation for distribution details.

Details

This function takes observed count data and covariates, then fits an MNM model using Nimble. The model estimates species-specific detection probabilities and abundances, allowing for covariate effects. The function also supports posterior predictive checks by comparing observed counts with predicted values.

Value

An MNM object that contains the following components:

- summary: Nimble model summary (mean, standard deviation, standard error, quantiles, effective sample size and Rhat value for all monitored values)
- n_parameters: Number of parameters in the model (for use in calculating information criteria)
- data: Observed abundances

- fitted_Y: Predicted values for Y. Posterior predictive checks can be performed by comparing fitted_Y with the observed data.
- logLik: Log-likelihood of the observed data (Y) given the model parameters.
- n_converged: Number of parameters with successful convergence (Rhat < 1.1).
- plot: traceplots and density plots for all monitored variables.

Note

Ensure that the dimensions of `Y`, `Xp`, and `Xn` match the requirements specified above. Mismatched dimensions will result in errors during model fitting.

References

- Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115.
- Mimnagh, N., Parnell, A., Prado, E., & Moral, R. D. A. (2022). Bayesian multi-species N-mixture models for unmarked animal communities. Environmental and Ecological Statistics, 29(4), 755-778.

See Also

`simulateData`: For generating example datasets compatible with this function.

Examples

```
# Example 1:
# Covariates must be of dimension (R, S, P1/P2). If covariates of an alternative dimension
# are used, they must first be coerced into the right format.
# If we have two abundance-covariates, one site-level covariate and one species-level
# covariate, they may be combined as follows:
R <- 10  # Number of sites
S <- 5   # Number of species
T<-5
Y <- array(sample(0:10, 100, replace = TRUE), dim = c(R, T, S))
covariate_1 <- runif(R)  # Site-level covariate
covariate_2 <- runif(S)  # Species-level covariate
# Expand covariate_1 to have S columns
expanded_covariate_1 <- matrix(rep(covariate_1, S), nrow = R, ncol = S)
# Expand covariate_2 to have R rows
expanded_covariate_2 <- t(matrix(rep(covariate_2, R), nrow = S, ncol = R))
# Combine into an array of dimensions (R, S, 2)
Xn <- array(c(expanded_covariate_1, expanded_covariate_2), dim = c(R, S, 2))
dim(Xn) # this is now in the correct format and can be used.
## Not run: result <- MNM(Y, Xn = Xn)
## Not run: print(result@summary)
#' data(birds_raw)
# Example 2: North American Breeding Bird Data
# Data must first be reformatted to an array of dimension (R,T,S,K)
R <- 24
T <- 10
S <- 20
K <- 6
# Ensure data is ordered consistently
birds_raw <- birds_raw[order(birds_raw$Route, birds_raw$Year, birds_raw$English_Common_Name), ]
# Create a 4D array with proper dimension
```

```
Y <- array(NA, dim = c(R, T, S, K))
# Map route, species, and year to indices
route_idx <- as.numeric(factor(birds_raw$Route))
species_idx <- as.numeric(factor(birds_raw$English_Common_Name))
year_idx <- as.numeric(factor(birds_raw$Year))
# Populate the array
stop_data <- as.matrix(birds_raw[, grep("^Stop", colnames(birds_raw))])
for (i in seq_len(nrow(birds_raw))) {
  Y[route_idx[i], , species_idx[i], year_idx[i]] <- stop_data[i, ]
  }
  # Assign dimnames
  dimnames(Y) <- list(
    Route = sort(unique(birds_raw$Route)),
      Stop = paste0("Stop", 1:T),
        Species = sort(unique(birds_raw$English_Common_Name)),
          Year = sort(unique(birds_raw$Year))
          )
# Selecting only 5 bird species and 1 year for analysis:
Y<-Y[,,1:5,1]
## Not run: model<-MNM_fit(Y=Y, AR=FALSE, Hurdle=FALSE, iterations=5000, burnin=1000)
```

MNM-class *Multi-Species N-Mixture (MNM) Model Class*

Description

The "MNM" class represents a multi-species N-mixture model object fitted using Nimble. It contains key model outputs, including parameter estimates, input data, predictions, log-likelihood, convergence diagnostics, and model summaries.

Slots

- `summary` A `data.frame` summarizing the model output, including posterior means, standard deviations, and convergence diagnostics (e.g., Rhat values).
- `n_parameters` A numeric value indicating the number of parameters estimated in the model.
- `estimates` Mean estimates for all monitored parameters.
- `fitted_Y` An array containing the fitted values (predicted responses) for the model.
- `data` An array containing the input data used to fit the model.
- `logLik` A numeric value representing the log-likelihood of the model.
- `n_converged` A numeric value indicating the number of parameters with Rhat < 1.1, showing convergence.
- `plot` traceplots and density plots for all monitored variables.

MNM_AR *Fit a Multi-Species N-Mixture Model with AR-1 Using Nimble*

Description

Fits a multi-species N-mixture (MNM) model with an autoregressive component (AR-1) using Nimble. This model is suitable for time-series data collected over long periods and includes covariates for abundance and detection probability.

Usage

```
MNM_AR(
  Y = NULL,
  iterations = 60000,
  burnin = 20000,
  thin = 10,
  Xp = NULL,
  Xn = NULL,
  ...
)
```

Arguments

`Y` Array of observed counts, with dimensions (R, T, S, K), where:

- R: Number of sites.
- T: Number of repeated counts (replicates).
- S: Number of species.
- K: Number of time points.

`iterations` Integer. Number of MCMC iterations to run. Defaults to 60,000.

`burnin` Integer. Number of iterations to discard as burn-in. Defaults to 20,000.

`thin` Integer. Thinning interval for saving MCMC samples. Defaults to 10.

`Xp` Array of detection covariates with dimensions `(R, S, P1)`, where:

- `R`: Number of sites.
- `S`: Number of species.
- `P1`: Number of detection probability covariates.

`Xn` Array of abundance covariates with dimensions `(R, S, K, P2)`, where:

- `R`: Number of sites.
- `S`: Number of species.
- `K`: Number of time points.
- `P2`: Number of abundance covariates.

`...` Additional arguments passed for prior distribution specification. Supported distributions include dnorm, dexp, dgamma, dbeta, dunif, dlnorm, dbern, dpois, dbinom, dcat, dmnorm, dwish, dchisq, dinvgamma, dt, dweib, ddirch, dmulti, dmvt. Default prior distributions are:

- prior_detection_probability: prior distribution for the detection probability intercept (`gamma`). Default is `'dnorm(0, 0.001)'`.

- prior_precision: prior distribution for the precision matrix for the species-level random effect. Default is `'dwish(Omega[1:S,1:S], df)'`.
- prior_mean: prior distribution for the mean of the species-level random effect (`mu`). Default is `'dnorm(0,0.001)'`.
- prior_hurdle: prior distribution for `theta`, the probability of structural zero in hurdle models. Default is `'dbeta(1,1)'`.
- prior_mean_AR: prior distribution for the mean of the autoregressive random effect (`phi`). Default is `'dnorm(0,0.001)'`.
- prior_sd_AR: prior distribution for the standard deviation of the autoregressive random effect (`phi`). Default is `'dexp(1)'`.

See Nimble (r-nimble.org) documentation for distribution details.

Details

The model incorporates temporal and species-level covariates and accounts for serial correlation in abundance using an AR(1) process. The input data should be structured as an array with dimensions (R,T,S,K). See parameter descriptions for details.

Features include:

- Posterior predictive checks via predicted vs observed abundance.
- Calculation of log-likelihood for model evaluation.
- Automatic monitoring of parameter convergence.

Value

An MNM object that contains the following components:

- summary: Nimble model summary (mean, standard deviation, standard error, quantiles, effective sample size and Rhat value for all monitored values).
- n_parameters: Number of parameters in the model (for use in calculating information criteria).
- data: Observed abundances.
- fitted_Y:Predicted values for Y: posterior predictive checks can be performed by comparing fitted_Y with the observed data.
- logLik:Log-likelihood of the observed data (Y) given the model parameters.
- n_converged: Number of parameters with successful convergence (Rhat < 1.1).
- plot: traceplots and density plots for all monitored variables.

Note

Ensure that the dimensions of `Y`, `Xp`, and `Xn` match the requirements specified above. Mismatched dimensions will result in errors during model fitting.

References

- Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115.
- Mimnagh, N., Parnell, A., Prado, E., & Moral, R. D. A. (2022). Bayesian multi-species N-mixture models for unmarked animal communities. Environmental and Ecological Statistics, 29(4), 755-778.

See Also

- `simulateData`: For generating example datasets compatible with this function.
- `MNM`: For details on creating covariate arrays Xp and Xn.

Examples

```
# Example 1:
Y <- array(rpois(1000, lambda = 10), dim = c(10, 10, 5, 2))
Xp <- array(runif(500), dim = c(10, 5, 2, 3))
Xn <- array(runif(1000), dim = c(10, 5, 2, 4))

# Fit the AR-1 model
## Not run: result <- MNM_AR(Y = Y, Xp = Xp, Xn = Xn)

# Check fitted vs observed abundance
## Not run: plot(result@data, result@fitted_Y)

#' data(birds_raw)

# Example 2: North American Breeding Bird Data
# Data must first be reformatted to an array of dimension (R,T,S,K)
R <- 24
T <- 10
S <- 20
K <- 6
# Ensure data is ordered consistently
birds_raw <- birds_raw[order(birds_raw$Route, birds_raw$Year, birds_raw$English_Common_Name), ]

# Create a 4D array with proper dimension
Y <- array(NA, dim = c(R, T, S, K))

# Map route, species, and year to indices
route_idx <- as.numeric(factor(birds_raw$Route))
species_idx <- as.numeric(factor(birds_raw$English_Common_Name))
year_idx <- as.numeric(factor(birds_raw$Year))

# Populate the array
stop_data <- as.matrix(birds_raw[, grep("^Stop", colnames(birds_raw))])

for (i in seq_len(nrow(birds_raw))) {
  Y[route_idx[i], , species_idx[i], year_idx[i]] <- stop_data[i, ]
  }

  # Assign dimnames
  dimnames(Y) <- list(
    Route = sort(unique(birds_raw$Route)),
      Stop = paste0("Stop", 1:T),
        Species = sort(unique(birds_raw$English_Common_Name)),
          Year = sort(unique(birds_raw$Year))
          )

# Selecting only 5 bird species  for analysis:
Y<-Y[,,1:5,]

## Not run: model<-MNM_fit(Y=Y, AR=TRUE, Hurdle=FALSE, iterations=10000, burnin=2000)
```

`MNM_control` *Generate Nimble Code for Multi-Species N-Mixture (MNM) Models*

Description

This function generates Nimble code for fitting various Multi-Species N-Mixture (MNM) models. These models can include standard MNM, hurdle components, autoregressive (AR) components, or both hurdle and AR components.

Usage

```
MNM_control(
  model = "MNM",
  prior_detection_probability = "dnorm(0, 0.001)",
  prior_precision = "dwish(Omega[1:S,1:S], df)",
  prior_mean = "dnorm(0,0.001)",
  prior_mean_AR = "dnorm(0,0.001)",
  prior_sd_AR = "dexp(1)",
  prior_hurdle = "dbeta(1,1)",
  Xp = NULL,
  Xn = NULL
)
```

Arguments

`model` Character. Specifies the type of MNM model to generate. Options are:

- `"MNM"`: Standard MNM model (default)
- `"Hurdle"`: Hurdle MNM model
- `"AR"`: Autoregressive MNM model
- `"HurdleAR"`: Hurdle MNM model with autoregression

`prior_detection_probability`
Character. Prior distribution for the detection probability intercept (`gamma`). Default is `'dnorm(0, 0.001)'`.

`prior_precision`
Character. Prior distribution for the precision matrix for the species-level random effect. Default is `'dwish(Omega[1:S,1:S], df)'`.

`prior_mean` Character. Prior distribution for the mean of the species-level random effect (`mu`). Default is `'dnorm(0,0.001)'`.

`prior_mean_AR` Character. Prior distribution for the mean of the autoregressive random effect (`phi`). Default is `'dnorm(0,0.001)'`.

`prior_sd_AR` Character. Prior distribution for the standard deviation of the autoregressive random effect (`phi`). Default is `'dexp(1)'`.

`prior_hurdle` Character. Prior distribution for `theta`, the probability of structural zero in hurdle models. Default is `'dbeta(1,1)'`.

`Xp` Array. Covariates influencing detection probability. Dimensions depend on the model:

- $R \times S \times P$ for models without AR
- $R \times S \times K \times P$ for AR models where:

- R: Number of sites
- S: Number of species
- K: Number of time points
- P: Number of covariates

`Xn` Array. Covariates influencing abundance. Dimensions depend on the model:

- $R \times S \times P$ for models without AR
- $R \times S \times K \times P$ for AR models

Details

The generated Nimble code can be used to fit Bayesian hierarchical MNM models. Users can specify prior distributions for key model parameters and provide covariate arrays that influence detection probability and abundance. The function validates prior specifications and adapts model code based on the selected model type. Supported models include:

- **MNM**: Standard MNM model
- **Hurdle**: MNM model with a hurdle component
- **AR**: MNM model with an autoregressive component
- **HurdleAR**: MNM model with both hurdle and AR components

This is an internal function, needed for implementing MNM models using the MNM_fit function. While the MNM_fit function allows the user to define prior distributions and linear covariates on detection probability and abundance, if the user wishes to implement more complex models in Nimble (via unsupported prior distributions or non-linear covariate effects), the Nimble model code may be extracted as in examples below, and modified by the user.

Value

Character. Nimble code for the specified MNM model, which can be used for further analysis or fitting.

Examples

```
# Example
# In order to implement scenarios involving nonlinear covariate
# effects or complex interaction terms,
# the user is invited to extract and modify the MNM Nimble code:
model<-MNM_control(model="MNM")
cat(model)
```

`MNM_fit` *Fit a Multi-Species N-Mixture (MNM) Model in Nimble*

Description

Fits a Multi-Species N-Mixture (MNM) model to observed count data using Nimble, with options to include autocorrelation (AR) and/or hurdle components for advanced ecological modeling.

Usage

```
MNM_fit(Y = NULL, AR = FALSE, Hurdle = FALSE, Xp = NULL, Xn = NULL, ...)
```

Arguments

`Y` Array of observed count data:

- Dimensions for standard MNM and hurdle models: $R \times T \times S$
- Dimensions for MNM with AR or both AR and hurdle components: $R \times T \times S \times K$
- `R`: Number of sites
- `T`: Number of replicates
- `S`: Number of species
- `K`: Number of time periods (required if `AR = TRUE`)

`AR` Logical. Indicates whether to include an autocorrelation component in the model. Defaults to `FALSE`.

`Hurdle` Logical. Indicates whether to include a hurdle component in the model. Defaults to `FALSE`.

`Xp` An array of covariates affecting detection probability, with dimensions (R, S, P1), where:

- `R`: Number of sites
- `S`: Number of species
- `P1`: Number of detection-related covariates See examples for implementation details.

`Xn` An array of covariates affecting abundance, with dimensions (R, S, P2), where:

- `R`: Number of sites
- `S`: Number of species
- `P2`: Number of abundance-related covariates

`...` Additional arguments for prior distributions. Supported priors include:

- `prior_detection_probability`, `prior_precision`, `prior_mean`, `prior_mean_AR`, `prior_sd_AR`, `prior_hurdle`.
- Supported distributions include: `dnorm`, `dexp`, `dgamma`, `dbeta`, `dunif`, `dlnorm`, `dbern`, `dpois`, `dbinom`, `dcat`, `dmnorm`, `dwish`, `dchisq`, `dinvgamma`, `dt`, `dweib`, `ddirch`, `dmulti`, `dmvt`. Refer to the Nimble documentation for details.

Details

This function implements the Bayesian MNM model to estimate latent species abundances and inter-species correlations based on observed count data. Extensions include options for incorporating autocorrelation (AR) to account for temporal dependencies and a hurdle model to handle zero inflation in the data. The input data and covariates must conform to specific dimensional requirements as described below.

The MNM model (Mimnagh et al., 2022) builds upon Royle's (2004) N-mixture model by allowing simultaneous modeling of multiple species, enabling inference about inter-species relationships and correlations.

Value

An MNM object that contains the following components:

- summary: Nimble model summary (mean, standard deviation, standard error, quantiles, effective sample size and Rhat value for all monitored values)

- n_parameters: Number of parameters in the model (for use in calculating information criteria)
- data: Observed abundances
- fitted_Y: Predicted values for Y. Posterior predictive checks can be performed by comparing fitted_Y with the observed data.
- logLik: Log-likelihood of the observed data (Y) given the model parameters.
- n_converged: Number of parameters with successful convergence (Rhat < 1.1).

Note

Ensure that the dimensions of `Y`, `Xp`, and `Xn` match the requirements specified above. Mismatched dimensions will result in errors during model fitting.#'

References

- Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115.
- Mimnagh, N., Parnell, A., Prado, E., & Moral, R. D. A. (2022). Bayesian multi-species N-mixture models for unmarked animal communities. Environmental and Ecological Statistics, 29(4), 755-778.

See Also

- `simulateData`: For generating example datasets compatible with this function.
- `MNM`: For details on creation of covariate arrays Xp and Xn.

Examples

```
# Example 1: Fit a standard MNM model
Y <- array(data = rpois(60, lambda = 5), dim = c(3, 5, 4))  # Simulated counts
Xp <- array(data = rnorm(60), dim = c(3, 4, 2))  # Detection covariates
Xn <- array(data = rnorm(60), dim = c(3, 4, 2))  # Abundance covariates

## Not run: model <- MNM_fit(Y = Y, AR = FALSE, Hurdle = FALSE, Xp = Xp, Xn = Xn)

#' # Example 2: Fit an MNM model with AR-1 component
Y <- array(data = rpois(180, lambda = 5), dim = c(3, 5, 4, 3))  # Simulated counts
Xp <- array(data = rnorm(180), dim = c(3, 4, 3, 2))  # Detection covariates
Xn <- array(data = rnorm(180), dim = c(3, 4, 3, 2))  # Abundance covariates

## Not run: model <- MNM_fit(Y = Y, AR = TRUE, Hurdle = FALSE, Xp = Xp, Xn = Xn)

# Example 3: Fit an MNM model with user-specified prior distributions
Y <- array(data = rpois(60, lambda = 5), dim = c(3, 5, 4))  # Simulated counts
Xp <- array(data = rnorm(60), dim = c(3, 4, 2))  # Detection covariates
Xn <- array(data = rnorm(60), dim = c(3, 4, 2))  # Abundance covariates

## Not run: model <- MNM_fit(Y = Y, AR = FALSE, Hurdle = TRUE, Xp = Xp, Xn = Xn,
                          prior_detection_probability="dnorm(0.01,0.01)")
## End(Not run)
# Access traceplots and density plots
## Not run: tracePlot(y, "N[10, 1]")
## Not run: density(y, "N[10, 1]")
```

MNM_Hurdle — *Fit a Multi-Species N-Mixture Model with Hurdle Component using Nimble*

Description

This function fits a multi-species N-mixture (MNM) model incorporating a Hurdle component to handle zero-inflated data, allowing for robust estimation of abundance and detection probabilities across multiple species and sites.

Usage

```
MNM_Hurdle(
  Y = NULL,
  iterations = 60000,
  burnin = 20000,
  thin = 10,
  Xp = NULL,
  Xn = NULL,
  ...
)
```

Arguments

`Y`	Array of observed counts, with dimensions (R, T, S, K), where: • R: Number of sites. • T: Number of repeated counts (replicates). • S: Number of species.
`iterations`	Integer. Number of iterations to be used in the JAGS model. Defaults to 60,000.
`burnin`	Integer. Number of iterations to be discarded as burn-in. Defaults to 20,000.
`thin`	Integer. Thinning interval for the MCMC chains. Defaults to 10.
`Xp`	Array of detection covariates with dimensions `(R, S, P1)`, where: • `R`: Number of sites. • `S`: Number of species. • `P1`: Number of detection probability covariates.
`Xn`	Array of abundance covariates with dimensions `(R, S, P2)`, where: • `R`: Number of sites. • `S`: Number of species. • `P2`: Number of abundance covariates.
`...`	Additional arguments passed for prior distribution specification. Supported distributions include dnorm, dexp, dgamma, dbeta, dunif, dlnorm, dbern, dpois, dbinom, dcat, dmnorm, dwish, dchisq, dinvgamma, dt, dweib, ddirch, dmulti, dmvt. Default prior distributions are: • prior_detection_probability: prior distribution for the detection probability intercept (`gamma`). Default is `'dnorm(0, 0.001)'`. • prior_precision: prior distribution for the precision matrix for the species-level random effect. Default is `'dwish(Omega[1:S,1:S], df)'`.

- prior_mean: prior distribution for the mean of the species-level random effect (`mu`). Default is `'dnorm(0,0.001)'`.
- prior_hurdle: prior distribution for `theta`, the probability of structural zero in hurdle models. Default is `'dbeta(1,1)'`.
- prior_mean_AR: prior distribution for the mean of the autoregressive random effect (`phi`). Default is `'dnorm(0,0.001)'`.
- prior_sd_AR: prior distribution for the standard deviation of the autoregressive random effect (`phi`). Default is `'dexp(1)'`.

See Nimble (r-nimble.org) documentation for distribution details.

Details

This function uses the Nimble framework to fit a Hurdle model, which combines a truncated Poisson distribution for non-zero counts with a separate process for modeling zero counts. The model is particularly suitable for ecological data with excess zeros, such as species occurrence data.

The model supports covariates influencing both abundance and detection probabilities, and outputs posterior distributions for model parameters, derived quantities, and predicted values. Convergence diagnostics and posterior predictive checks can also be performed using the returned results.

Value

An MNM object that contains the following components:

- summary: Nimble model summary statistics (mean, standard deviation, standard error, quantiles, effective sample size and Rhat value for all monitored values)
- n_parameters: Number of parameters in the model (for use in calculating information criteria).
- data: Observed abundances.
- fitted_Y: Predicted values for Y. Posterior predictive checks can be performed by comparing fitted_Y with the observed data.
- logLik: Log-likelihood of the observed data (Y) given the model parameters.
- n_converged: Number of parameters with successful convergence (Rhat < 1.1).
- plot: traceplots and density plots for all monitored variables.

Note

Ensure that the dimensions of `Y`, `Xp`, and `Xn` match the requirements specified above. Mismatched dimensions will result in errors during model fitting.

References

- Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115.
- Mimnagh, N., Parnell, A., Prado, E., & Moral, R. D. A. (2022). Bayesian multi-species N-mixture models for unmarked animal communities. Environmental and Ecological Statistics, 29(4), 755-778.

See Also

- `simulateData`: For generating example datasets compatible with this function.
- `MNM`: For details on creating covariate arrays Xp and Xn.

Examples

```
# Example 1:
Y <- array(rpois(100, lambda = 5), dim = c(10, 5, 2))
Xp <- array(runif(100), dim = c(10, 2, 5))
Xn <- array(runif(100), dim = c(10, 2, 3))

## Not run: model <- MNM_Hurdle(Y = Y, Xp = Xp, Xn = Xn)

# Accessing results
## Not run: print(model@summary)

#' data(birds_raw)

# Example 2: North American Breeding Bird Data
# Data must first be reformatted to an array of dimension (R,T,S,K)
R <- 24
T <- 10
S <- 20
K <- 6
# Ensure data is ordered consistently
birds_raw <- birds_raw[order(birds_raw$Route, birds_raw$Year, birds_raw$English_Common_Name), ]

# Create a 4D array with proper dimension
Y <- array(NA, dim = c(R, T, S, K))

# Map route, species, and year to indices
route_idx <- as.numeric(factor(birds_raw$Route))
species_idx <- as.numeric(factor(birds_raw$English_Common_Name))
year_idx <- as.numeric(factor(birds_raw$Year))

# Populate the array
stop_data <- as.matrix(birds_raw[, grep("^Stop", colnames(birds_raw))])

for (i in seq_len(nrow(birds_raw))) {
  Y[route_idx[i], , species_idx[i], year_idx[i]] <- stop_data[i, ]
  }

  # Assign dimnames
  dimnames(Y) <- list(
    Route = sort(unique(birds_raw$Route)),
      Stop = paste0("Stop", 1:T),
        Species = sort(unique(birds_raw$English_Common_Name)),
          Year = sort(unique(birds_raw$Year))
          )

# Selecting only 5 bird species and 1 year for analysis:
Y<-Y[,,1:5,1]

## Not run: model<-MNM_fit(Y=Y, AR=FALSE, Hurdle=TRUE, iterations=5000, burnin=1000)
```

`MNM_Hurdle_AR` *Fit a multi-species N-mixture (MNM) Hurdle-AR model using Nimble*

Description

Fits a multi-species N-mixture model (MNM) with an autoregressive (AR-1) component and a Hurdle (zero-altered) component using Nimble. This model is suitable for zero-inflated data and data collected over extended time periods.

Usage

```
MNM_Hurdle_AR(
  Y = NULL,
  iterations = 60000,
  burnin = 20000,
  thin = 10,
  Xp = NULL,
  Xn = NULL,
  ...
)
```

Arguments

`Y`	Array of observed counts with dimensions `(R, T, S, K)`, where: • `R`: Number of sites. • `T`: Number of replicates. • `S`: Number of species. • `K`: Number of time periods.
`iterations`	Number of MCMC iterations for model fitting. Default is `60,000`.
`burnin`	Number of initial iterations to discard as burn-in. Default is `20,000`.
`thin`	Thinning interval for the MCMC chains. Default is `10`.
`Xp`	Array of detection covariates with dimensions `(R, S, P1)`, where: • `R`: Number of sites. • `S`: Number of species. • `P1`: Number of detection probability covariates.
`Xn`	Array of abundance covariates with dimensions `(R, S, K, P2)`, where: • `R`: Number of sites. • `S`: Number of species. • `K`: Number of time points. • `P2`: Number of abundance covariates.
`...`	Additional arguments passed for prior distribution specification. Supported distributions include dnorm, dexp, dgamma, dbeta, dunif, dlnorm, dbern, dpois, dbinom, dcat, dmnorm, dwish, dchisq, dinvgamma, dt, dweib, ddirch, dmulti, dmvt. Default prior distributions are: • prior_detection_probability: prior distribution for the detection probability intercept (`gamma`). Default is `'dnorm(0, 0.001)'`. • prior_precision: prior distribution for the precision matrix for the species-level random effect. Default is `'dwish(Omega[1:S,1:S], df)'`. • prior_mean: prior distribution for the mean of the species-level random effect (`mu`). Default is `'dnorm(0,0.001)'`. • prior_hurdle: prior distribution for `theta`, the probability of structural zero in hurdle models. Default is `'dbeta(1,1)'`.

- prior_mean_AR: prior distribution for the mean of the autoregressive random effect (`phi`). Default is `'dnorm(0,0.001)'`.
- prior_sd_AR: prior distribution for the standard deviation of the autoregressive random effect (`phi`). Default is `'dexp(1)'`.

See Nimble (r-nimble.org) documentation for distribution details.

Details

The MNM_Hurdle_AR model extends the standard N-mixture model by incorporating:

- **Hurdle (zero-altered) component:** Handles zero-inflated data by modelling excess zeros separately.
- **Autoregressive (AR-1) component:** Accounts for temporal dependencies in abundance data.

The model is fitted to data formatted as produced by the `simulateData` function. Covariates affecting detection probability and abundance may also be provided. Results include posterior summaries, model diagnostics, and predictions for posterior predictive checks.

Value

An object of class `MNM` with the following components:

- summary: Summary statistics for monitored parameters, including mean, standard deviation, standard error, quantiles, effective sample size, and Rhat values.
- n_parameters: Number of parameters in the model (useful for calculating information criteria).
- data: Observed abundances (`Y`).
- fitted_Y: Predicted values for `Y`. Use these for posterior predictive checks by comparing them with observed data.
- logLik: Log-likelihood of the observed data (`Y`) given the model parameters.
- n_converged: Number of parameters with successful convergence (Rhat < 1.1).
- plot: traceplots and density plots for all monitored variables.

Note

Ensure that the dimensions of `Y`, `Xp`, and `Xn` match the requirements specified above. Mismatched dimensions will result in errors during model fitting.

References

- Royle, J. A. (2004). N-mixture models for estimating population size from spatially replicated counts. Biometrics, 60(1), 108-115.
- Mimnagh, N., Parnell, A., Prado, E., & Moral, R. D. A. (2022). Bayesian multi-species N-mixture models for unmarked animal communities. Environmental and Ecological Statistics, 29(4), 755-778.

See Also

- `simulateData`: For generating example datasets compatible with this function.
- `MNM`: For details on creating covariate arrays Xp and Xn.

Examples

```
# Example 1: Simulate data and fit the model
# Simulating example data
set.seed(42)
R <- 5  # Number of sites
T <- 10  # Number of replicates
S <- 3  # Number of species
K <- 2  # Number of time periods
P1 <- 2  # Number of detection covariates
P2 <- 3  # Number of abundance covariates

x<-simulateData(model="HurdleAR", R=R, T=T, ,S=S, K=K)
Xp <- array(runif(R * S * K * P1), dim = c(R, S, K, P1))
Xn <- array(runif(R * S * K * P2), dim = c(R, S, K, P2))
# Fit the MNM_Hurdle_AR model
## Not run: result <- MNM_Hurdle_AR(Y = x[["Y"]], Xp = Xp, Xn = Xn)
# Access results
## Not run: print(result@summary)

#' data(birds_raw)

# Example 2: North American Breeding Bird Data
# Data must first be reformatted to an array of dimension (R,T,S,K)
R <- 24
T <- 10
S <- 20
K <- 6
# Ensure data is ordered consistently
birds_raw <- birds_raw[order(birds_raw$Route, birds_raw$Year, birds_raw$English_Common_Name), ]

# Create a 4D array with proper dimension
Y <- array(NA, dim = c(R, T, S, K))

# Map route, species, and year to indices
route_idx <- as.numeric(factor(birds_raw$Route))
species_idx <- as.numeric(factor(birds_raw$English_Common_Name))
year_idx <- as.numeric(factor(birds_raw$Year))

# Populate the array
stop_data <- as.matrix(birds_raw[, grep("^Stop", colnames(birds_raw))])

for (i in seq_len(nrow(birds_raw))) {
  Y[route_idx[i], , species_idx[i], year_idx[i]] <- stop_data[i, ]
  }

  # Assign dimnames
  dimnames(Y) <- list(
    Route = sort(unique(birds_raw$Route)),
      Stop = paste0("Stop", 1:T),
        Species = sort(unique(birds_raw$English_Common_Name)),
          Year = sort(unique(birds_raw$Year))
          )

# Selecting only 5 bird species for analysis:
Y<-Y[,,1:5,]
```

```
## Not run: model<-MNM_fit(Y=Y, AR=TRUE, Hurdle=TRUE, iterations=5000, burnin=1000)
```

`predictY` *Predict Fitted Values for "MNM" Class*

Description

Returns the predicted (fitted) values, denoted as Y, from an object of class `"MNM"`.

Usage

```
predictY(object)

## S4 method for signature 'MNM'
predictY(object)
```

Arguments

`object`	An object of class `"MNM"`.

Value

An array containing the predicted (fitted) values from the model.

Methods (by class)

- `predictY(MNM)`: Predict fitted values for the "MNM" class.

`simulateData` *Simulate Data for Multi-Species N-Mixture Models*

Description

Simulates multi-species correlated abundance data for various Multi-Species N-Mixture (MNM) model types, including standard MNM, Hurdle, AR (autoregressive), and HurdleAR models.

Usage

```
simulateData(
  model = "MNM",
  R = 10,
  S = 2,
  T = 5,
  prob = "all",
  abundance = "small",
  K = 4,
  theta = 0.5
)
```

Arguments

`model`	Character. Specifies the model type. Options are `"MNM"`, `"Hurdle"`, `"AR"`, `"HurdleAR"`. Default is `"MNM"`.
`R`	Integer. Number of sites. Default is `10`.
`S`	Integer. Number of species. Default is `2`.
`T`	Integer. Number of replicates. Default is `5`.
`prob`	Character. Specifies the range of detection probabilities: • `"small"`: Detection probabilities < 0.4. • `"large"`: Detection probabilities > 0.5. • `"all"`: Detection probabilities between 0.01 and 0.99 (default).
`abundance`	Character. Specifies the abundance size: • `"small"`: Latent species abundance between 0 and 50. • `"large"`: Latent species abundance between 0 and 700. Default is `"small"`.
`K`	Integer. Number of time points (used for AR models). Default is `4`.
`theta`	Numeric. Probability of zero-inflation (used for hurdle models). Default is `0.5`.

Details

This function generates abundance data for multi-species N-mixture models under different configurations:

- **MNM**: Standard multi-species N-mixture model.
- **Hurdle**: Includes a hurdle component to model zero-inflated data.
- **AR**: Includes an autoregressive (AR) component for temporal dependencies.
- **HurdleAR**: Combines hurdle and AR components for zero-inflation and temporal dependencies. The output includes observed and true abundances, detection probabilities, latent variables, and covariance information for the random effects.

Value

A list containing:

- **Y**: Array of observed abundances.
- **N**: Array of true abundances.
- **p**: Array of detection probabilities.
- **Sigma**: Covariance matrix for the multivariate normal variable `a`.
- **mu**: Mean vector for the multivariate normal variable `a`.
- **lambda**: Latent abundance rate parameter.
- **correlation**: Correlation matrix derived from `Sigma`.
- **R, T, S, K**: Number of sites, sampling occasions, species, and time points.
- Additional elements depending on the model type:
  - **phi**: Autoregression parameter (AR and HurdleAR models).
  - **muPhi**: Mean of the autoregressive parameter (AR and HurdleAR models).
  - **varPhi**: Variance of the autoregressive parameter (AR and HurdleAR models).
  - **zeros**: Matrix of zero-indicators for hurdle models.
  - **theta**: Zero-inflation parameter for hurdle models.

See Also

- `simulateData_MNM`: Helper function for simulating standard MNM data.
- `simulateData_Hurdle`: Helper function for simulating hurdle MNM data.
- `simulateData_AR`: Helper function for simulating AR MNM data.
- `simulateData_Hurdle_AR`: Helper function for simulating hurdle AR MNM data.

Examples

```
# Simulate data for a standard MNM model
data <- simulateData(model = "MNM", R = 10, S = 3, T = 5, prob = "all",
abundance = "small")

# Simulate data for a hurdle model
data <- simulateData(model = "Hurdle", R = 10, S = 3, T = 5, prob = "large",
abundance = "large", theta = 0.3)

# Simulate data for an autoregressive model
data <- simulateData(model = "AR", R = 10, S = 2, T = 5, K = 4, prob = "small",
abundance = "small")

# Simulate data for a hurdle autoregressive model
data <- simulateData(model = "HurdleAR", R = 10, S = 3, T = 5, K = 4, prob = "all",
abundance = "large", theta = 0.5)
```

`tracePlot` *Trace Plot Generic and Method for "MNM" Class*

Description

The `tracePlot` function generates trace plots for parameters in objects. This documentation includes the generic and the method for the `"MNM"` class.

Usage

```
tracePlot(x, param, ...)

## S4 method for signature 'MNM'
tracePlot(x, param, ...)
```

Arguments

`x`	An object of class `"MNM"`.
`param`	The name of the parameter to plot (e.g., `"N[8, 1]"`).
`...`	Additional arguments (not used for `"MNM"`).

Value

A trace plot for the specified parameter is displayed.

Methods (by class)

- `tracePlot(MNM)`: Method for objects of class `"MNM"`.

Examples

```
## Not run:
# Assuming `y` is an object of class "MNM" with plots stored
tracePlot(y, "N[8, 1]")  # Generates a trace plot for parameter N[8, 1]

## End(Not run)
```

`validate_all_priors` *Validate User-Specified Priors for Nimble Models*

Description

Validates a set of user-defined priors to ensure that they specify valid distributions supported by Nimble and have the correct parameters for each distribution.

Usage

```
validate_all_priors(priors)
```

Arguments

`priors`	A named list of priors specified as strings, where the name is the prior name (e.g., `prior_mean`) and the value is the prior distribution (e.g., `'dnorm(0, 0.001)'`).

Details

The function checks the following:

- Whether the specified distribution is supported by Nimble.
- Whether the correct number of parameters is provided for the distribution.

If any prior is invalid, the function throws an informative error with details about the issue.

Value

Returns `TRUE` if all priors are valid. Throws an error if any prior is invalid.

References

- NIMBLE Development Team (2021). NIMBLE: An R Package for Programming with BUGS Models. `https://r-nimble.org/`

See Also

- Nimble documentation for a full list of supported distributions.

`validate_prior` *Validate User-Specified Prior for a Single Prior*

Description

Validates a single user-defined prior to ensure it specifies a valid distribution supported by Nimble and has the correct parameters for the distribution.

Usage

```
validate_prior(prior_name, prior_string)
```

Arguments

`prior_name`	A string representing the name of the prior (e.g., `prior_mean`).
`prior_string`	A string specifying the prior distribution (e.g., `'dnorm(0, 0.001)'`).

Value

Returns `TRUE` if the prior is valid. Throws an error if the prior is invalid.

Examples

```
validate_prior("prior_mean", "dnorm(0, 0.001)")
```

Index