Package 'Hmsc'

August 29, 2025

Title Hierarchical Model of Species Communities

Type Package

Version 3.3-7

Description Hierarchical Modelling of Species Communities (HMSC) is a model-based approach for analyzing community ecological data. This package implements it in the Bayesian framework with Gibbs Markov chain Monte Carlo (MCMC) sampling (Tikhonov et al. (2020) <doi:10.1111/2041-210X.13345>).

License GPL-3 | file LICENSE

URL https://www.helsinki.fi/en/researchgroups/statistical-ecology/software/hmsc

BugReports https://github.com/hmsc-r/HMSC/issues/

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests R.rsp, testthat, corrplot

VignetteBuilder R.rsp

Depends R (>= 3.0.2), coda

Imports abind, ape, BayesLogit, fields, FNN, ggplot2, MASS, Matrix, matrixStats, MCMCpack, methods, nnet, rlang, parallel, pracma, pROC, sp, statmod, truncnorm

NeedsCompilation no

Author Gleb Tikhonov [aut], Otso Ovaskainen [aut, cre] (ORCID: <https://orcid.org/0000-0001-9750-4421>), Jari Oksanen [aut], Melinda de Jonge [aut], Oystein Opedal [aut], Tad Dallas [aut]

Maintainer Otso Ovaskainen <otso.t.ovaskainen@jyu.fi>

Repository CRAN

Date/Publication 2025-08-28 22:30:16 UTC

Contents

alignPosterior *alignPosterior*

Description

Aligns posterior in terms of variables susceptible to label switching

Usage

```
alignPosterior(hM)
```

Arguments

hM	a fitted Hmsc model object

Value

an Hmsc model object that is identical to the input except the posterior being aligned

Examples

```
# Align the posterior for a previously fitted HMSC model
m = alignPosterior(TD$m)
```

biPlot *biPlot*

Description

Constructs an ordination biplot based on the fitted model

Usage

```
biPlot(
  hM,
  etaPost,
  lambdaPost,
  factors = c(1, 2),
  colVar = NULL,
  colors = NULL,
  spNames = hM$spNames,
  ...
)
```

Arguments

hM	a fitted Hmsc model object
etaPost	posterior distribution of site loadings (Eta)
lambdaPost	posterior distribution of species loadings (Lambda)
factors	indices of the two factors to be plotted
colVar	the environmental covariate from XData according to which the sites are to be coloured
colors	controls the colors of the heatmap. For continuous covariates, colors should be given as a name of palette, with default value colorRampPalette(c("blue","white","red")), or as a vector of colours. For factors, colors should be given as a vector of colours, e.g. c("blue","red").
spNames	a vector of species names to be added to the ordination diagram
...	other parameters passed to the function.

Examples

```
# Construct an ordination biplot using two chosen latent factors from a previously fitted HMSC model
etaPost = getPostEstimate(TD$m, "Eta")
lambdaPost=getPostEstimate(TD$m, "Lambda")
biPlot(TD$m, etaPost=etaPost, lambdaPost=lambdaPost, factors=c(1,2))
```

---

c.Hmsc *Combine Posterior Samples of Several Hmsc Models*

Description

Function combines posterior samples of several sampled Hmsc models (see sampleMcmc) as new chains in the first fitted model. The combined models must be comparable, and there are some tests for detecting non-equal models. These tests will only give warning, and it is at user deliberation to decide which models and which posterior samples can be combined. You should be careful not start two models from the same random number seed, because these will only duplicate your data instead of providing new independent samples.

Usage

```
## S3 method for class 'Hmsc'
c(...)
```

Arguments

...	Sampled Hmsc models with posterior samples that will be added as new chaings in the first listed model.

Value

An Hmsc model with chains of posterior samples.

Examples

```
## Fit a toy model with two chains
m1 <- sampleMcmc(TD$m, samples=10, transient=5, nChains=2, verbose=0)
## Need more data? Add chains: check carefully that these are
## sampled exactly like the previous model
m2 <- sampleMcmc(TD$m, nChains=2, samples=10, transient=5, verbose=0)
## Now four chains
m4 <- c(m1, m2)
m4
```

---

computeAssociations *computeAssociations*

Description

Computes the species association matrices associated with each random level

Usage

```
computeAssociations(hM, start = 1, thin = 1)
```

Arguments

hM	a fitted Hmsc model object
start	index of first MCMC sample included
thin	thinning interval of posterior distribution

Value

list of association matrices (ω) corresponding to each random level in the model

Examples

```
# Compute the associations (residual correlations) between species from a HMSC model
assoc = computeAssociations(TD$m)
```

computeDataParameters *computeDataParameters*

Description

Computes initial values before the sampling starts

Usage

```
computeDataParameters(hM, compactFormat = FALSE)
```

Arguments

hM	a fitted Hmsc model object
compactFormat	disable computation of 3D arrays for phylogeny and spatial priors. Only used internally for reduced size of exported object to Hmsc-HPC.

Value

a list including pre-computed matrix inverses and determinants (for phylogenetic and spatial random effects) needed in MCMC sampling

computeInitialParameters *computeInitialParameters*

Description

Computes initial parameter values before the sampling starts

Usage

```
computeInitialParameters(hM, initPar, computeZ = TRUE)
```

Arguments

hM	a Hmsc model object
initPar	a list of initial parameter values
computeZ	whether to compute the latent predictor Z. Disable only for the purpose of Hmsc-HPC initialization.

Value

a list of Hmsc model parameters

computePredictedValues *computePredictedValues*

Description

Computes predicted values from the fitted Hmsc model

Usage

```
computePredictedValues(
  hM,
  partition = NULL,
  partition.sp = NULL,
  start = 1,
  thin = 1,
  Yc = NULL,
  mcmcStep = 1,
  expected = TRUE,
  initPar = NULL,
  nParallel = 1,
  nChains = length(hM$postList),
  updater = list(),
  verbose = hM$verbose,
  alignPost = TRUE
)

pcomputePredictedValues(
  hM,
  partition = NULL,
  partition.sp = NULL,
  start = 1,
  thin = 1,
  Yc = NULL,
  mcmcStep = 1,
  expected = TRUE,
  initPar = NULL,
  nParallel = 1,
  useSocket = .Platform$OS.type == "windows",
  nChains = length(hM$postList),
  updater = list(),
  verbose = nParallel == 1,
  alignPost = TRUE
)
```

Arguments

hM	a fitted Hmsc model object
partition	partition vector for cross-validation created by createPartition
partition.sp	species partitioning vector for conditional cross-validation
start	index of first MCMC sample included
thin	thinning interval of posterior distribution
Yc	response matrix on which the predictions are to be conditioned
mcmcStep	number of MCMC steps used to make conditional predictions
expected	whether expected values (TRUE) or realizations (FALSE) are to be predicted
initPar	a named list of parameter values used for initialization of MCMC states
nParallel	number of parallel processes by which the chains are executed
nChains	number of independent MCMC chains to be run
updater	a named list, specifying which conditional updaters should be omitted
verbose	the interval between MCMC steps printed to the console
alignPost	boolean flag indicating whether the posterior of each chains should be aligned
useSocket	(logical) use socket clusters in parallel processing; these can be used in all operating systems, but they are usually slower than forking which can only be used in non-Windows operating systems (macOS, Linux, unix-like systems).

Details

There are two alternative functions computePredictedValues and pcomputePredictedValues. Function pcomputePredictedValues uses more aggressive parallelization and can be much faster when partition is used. Function computePredictedValues can run chains of each sampleMcmc partition in parallel, but pcomputePredictedValues can run each partition fold times chain in parallel (if hardware and operating systems permit). Function pcomputePredictedValues is still experimental, and therefore we provide both the old and new functions, but the old functions is scheduled to be removed in the future. Species partitions are not yet parallelized, and they can be very slow, especially with many mcmcSteps.

If the option partition is not used, the posterior predictive distribution is based on the model fitted to the full data. If the option partition is used but partition.sp is not used, the posterior predictive distribution is based on cross-validation over the sampling units. If partition.sp is additionally used, then, when predictions are made for each fold of the sampling units, the predictions are done separately for each fold of species. When making the predictions for one fold of species, the predictions are conditional on known occurrences (those observed in the data) of the species belonging to the other folds. If partition.sp is used, the parameter mcmcStep should be set high enough to obtain appropriate conditional predictions. The option Yc can be used alternatively to partition.sp if the conditioning is to be done based on a fixed set of data (independently of which sampling unit and species the predictions are made for).

Value

an array of model predictions, made for each posterior sample

See Also

predict.Hmsc

Examples

```
# Compute predicted values using a previously fitted HMSC model
preds = computePredictedValues(TD$m)

## Not run:
# Compute predicted values for a previously fitted HMSC model using 2 folds
partition = createPartition(TD$m, nfolds = 2)
predsCV1 = computePredictedValues(TD$m,partition=partition)

# Compute conditional predictions for a previously fitted HMSC model using 2 folds
partition = createPartition(TD$m, nfolds = 2)
predsCV2 = computePredictedValues(TD$m, partition = partition,
partition.sp = 1:TD$m$ns, mcmcStep = 100)

## End(Not run)
```

---

computeSAIR *computeSAIR*

Description

Computes the shared and idiosyncratic responses to measured and latent predictors

Usage

```
computeSAIR(hM, X = NULL)
```

Arguments

hM	a fitted Hmsc model object
X	a design matrix to be used in the computations (as defaul hM$X)

Details

The shared and idiosyncratic responses are computed only for models without traits.

Value

returns the posterior distribution of the parameters mu.X2,V.XX,mu.Omega2,V.OmegaOmega,mu.tot,V.tot,s described Ovaskainen and Abrego (manuscript): Measuring niche overlap with joint species distribution models: shared and idiosyncratic responses of the species to measured and latent predictors

Examples

```
# Simulate a small dataset, fit Hmsc model to it, compute SAIR, and show posterior means

nc = 2
ns = 5
ny = 10
mu = rnorm(n = nc)
X = matrix(rnorm(n=nc*ny),nrow=ny)
X[,1] = 1
eps = matrix(rnorm(nc*ns),nrow=nc)
L = matrix(rep(X%*%mu,ns),nrow=ny)  + X%*%eps
Y = pnorm(L)
m = Hmsc(Y = Y, XData = data.frame(env = X[,2]), distr = "probit")
m = sampleMcmc(m,samples=100,transient=50,verbose = 0)
SI = computeSAIR(m)
colMeans(SI)
```

`computeVariancePartitioning` *computeVariancePartitioning*

Description

Computes variance components with respect to given grouping of fixed effects and levels of random effects

Usage

```
computeVariancePartitioning(
  hM,
  group = NULL,
  groupnames = NULL,
  start = 1,
  na.ignore = FALSE
)
```

Arguments

`hM`	a fitted `Hmsc` model object
`group`	vector of numeric values corresponding to group identifiers in groupnames. If the model was defined with `XData` and `XFormula`, the default is to use model terms.
`groupnames`	vector of names for each group of fixed effect. Should match `group`. If the model was defined with `XData` and `XFormula`, the default is to use the labels of model terms.
`start`	index of first MCMC sample included
`na.ignore`	logical. If TRUE, covariates are ignored for sites where the focal species is NA when computing variance-covariance matrices for each species

Details

The vector group has one value for each column of the matrix hM$X, describing the index of the group in which this column is to be included. The names of the group are given by groupnames. The output object VP$vals gives the variance proportion for each group and species. The output object VP$R2T gives the variance among species explained by traits, measured for species' responses to covariates (VP$R2T$Beta) and species occurrences (VP$R2T$Y)

Value

returns an object VP with components VP$vals, VP$R2T, VP$group and VP$groupnames.

See Also

Use plotVariancePartitioning to display the result object.

Examples

```
# Partition the explained variance for a previously fitted model
# without grouping environmental covariates
VP = computeVariancePartitioning(TD$m)

# Partition the explained variance for a previously fitted model
# while grouping the two environmental variables together
VP = computeVariancePartitioning(TD$m, group=c(1,1), groupnames = c("Habitat"))
```

---

computeWAIC *computeWAIC*

Description

Computes the value of WAIC (Widely Applicable Information Criterion) for the Hmsc model

Usage

```
computeWAIC(hM, ghN = 11, byColumn = FALSE)
```

Arguments

hM	a fitted Hmsc model object
ghN	order of Gauss-Hermite quadrature for approximate numerical integration
byColumn	describes whether WAIC is computed for the entire model byColumn=FALSE or for each column (i.e. species) byColumn=TRUE

Details

The result is exact for normal and probit observational models. For Poisson-type observational model the result is obtained through numerical integration using Gauss-Hermite quadrature.

Value

the scalar WAIC

Examples

```
# Compute WAIC of previously sampled Hmsc object
WAIC = computeWAIC(TD$m)
```

---

constructGradient *constructGradient*

Description

Constructs an environmental gradient over one of the variables included in XData

Usage

```
constructGradient(
  hM,
  focalVariable,
  non.focalVariables = list(),
  ngrid = 20,
  coordinates = list()
)
```

Arguments

hM	a fitted Hmsc model object
focalVariable	focal variable over which the gradient is constructed
non.focalVariables	list giving assumptions on how non-focal variables co-vary with the focal variable or a single number given the default type for all non-focal variables
ngrid	number of points along the gradient (for continuous focal variables)
coordinates	A named list of coordinates were model is evaluated in spatial or temporal models. The name should be one of the random levels, and value can be "c" for mean of coordinates (default), "i" for infinite coordinates without effect of spatial dependence, or a numeric vector of coordinates where the model is evaluated.

Details

In basic form, non.focalVariables is a list, where each element is on the form variable=list(type,value), where variable is one of the non-focal variables, and the value is needed only if type = 3. Alternatives type = 1 sets the values of the non-focal variable to the most likely value (defined as expected value for covariates, mode for factors), type = 2 sets the values of the non-focal variable

to most likely value, given the value of focal variable, based on a linear relationship, and `type = 3` fixes to the value given. As a shortcut, a single number `1` or `2` can be given as a type used for all non-focal variables. If a `non.focalVariable` is not listed, `type=2` is used as default. Note that if the focal variable is continuous, selecting type 2 for a non-focal categorical variable can cause abrupt changes in response.

The function needs access to the original `XData` data frame, and cannot be used if you defined your model with `X` model matrix. In that case you must construct your gradient manually.

Value

a named list with slots `XDataNew`, `studyDesignNew` and `rLNew`

See Also

`plotGradient`, `predict`.

Examples

```
# Construct gradient for environmental covariate called 'x1'.
Gradient = constructGradient(TD$m, focalVariable="x1")

# Construct gradient for environmental covariate called 'x1'
# while setting the other covariate to its most likely values
Gradient = constructGradient(TD$m, focalVariable="x1",non.focalVariables=list(x2=list(1)))
```

`constructKnots` *constructKnots*

Description

Construct a Regular Grid of Knot Locations for Spatial GPP Model

Usage

```
constructKnots(sData, nKnots = NULL, knotDist = NULL, minKnotDist = NULL)
```

Arguments

`sData`	a dataframe containing spatial or temporal coordinates of units of the random level
`nKnots`	the number of knots wanted on the spatial dimension with the shortest range
`knotDist`	the distance between the wanted knots
`minKnotDist`	the minimum distance of a knot to the nearest data point

Details

This is a helper function for spatial Hmsc models with the spatial method set to GPP where user must provide knot locations. Knot locations with a distance greater than `minKnotDist` to the nearest data point are dropped from the grid. If the input locations are `SpatialPoints` data, these are treated like Euclidean coordinates, and if the points are not projected, a warning is issued.

Only one of `nKnots` and `minKnotDist` arguments can be provided.

Value

a data frame with knot locations

Examples

```
#Creating knots for some 2 dimensional spatial data
n = 100
xycoords = matrix(runif(2*n),ncol=2)
xyKnots = constructKnots(xycoords,knotDist = 0.2, minKnotDist = 0.5)
```

`convertToCodaObject` *convertToCodaObject*

Description

Converts the Hmsc posterior into a named list of mcmc.list objects

Usage

```
convertToCodaObject(
  hM,
  start = 1,
  spNamesNumbers = c(TRUE, TRUE),
  covNamesNumbers = c(TRUE, TRUE),
  trNamesNumbers = c(TRUE, TRUE),
  Beta = TRUE,
  Gamma = TRUE,
  V = TRUE,
  Sigma = TRUE,
  Rho = TRUE,
  Eta = TRUE,
  Lambda = TRUE,
  Alpha = TRUE,
  Omega = TRUE,
  Psi = TRUE,
  Delta = TRUE
)
```

Arguments

hM	a fitted Hmsc model object
start	index of first MCMC sample included
spNamesNumbers	logical of length 2, where first entry controls whether species names are printed, and second entry controls whether species numbers are printed
covNamesNumbers	Logical of length 2, where first entry controls whether covariate names are printed, and second entry controls whether covariate numbers are printed
trNamesNumbers	Logical of length 2, where first entry controls whether trait names are printed, and second entry controls whether traits numbers are printed
Beta	logical indicating whether posterior of Beta is included
Gamma	logical indicating whether posterior of Gamma is included
V	logical indicating whether posterior of V is included
Sigma	logical indicating whether posterior of Sigma is included
Rho	logical indicating whether posterior of Rho is included
Eta	logical indicating whether posterior of Eta is included
Lambda	logical indicating whether posterior of Lambda is included
Alpha	logical indicating whether posterior of Alpha is included
Omega	logical indicating whether posterior of Omega is included
Psi	logical indicating whether posterior of Psi is included
Delta	logical indicating whether posterior of Delta is included

Value

A named list that can be analysed with **coda** functions.

Examples

```
# Convert recorded posterior samples in \code{Hmsc} object to coda object
codaObject = convertToCodaObject(TD$m)

# Convert recorded posterior samples, starting from sample 100, in m object to coda object
codaObject = convertToCodaObject(TD$m, start=100)
```

coralCombine *coralCombine*

Description

Extracts CORAL-like summary from backbone fitted Hmsc-class object and combines with CORAL models

Usage

```
coralCombine(m, muList.coral, VList.coral)
```

Arguments

m	fitted Hmsc-class object
muList.coral	matrix of CORAL posterior means
VList.coral	matrix of CORAL posterior flattened variances

Value

list with combined means and covariance matrices

coralGetRareSpeciesPriors *coralGetRareSpeciesPriors*

Description

Constructs CORAL priors

Usage

```
coralGetRareSpeciesPriors(
  m,
  spNames.rare,
  TrData.rare = NULL,
  phyloTree = NULL,
  C.common.rare = NULL,
  interceptVarInflation = 5
)
```

Arguments

m	fitted Hmsc-class object
spNames.rare	vector of species names that are considered rare
TrData.rare	dataframe of traits for rare species, if non-trivial traits were used in the backbone Hmsc model
phyloTree	phylogeny tree covering both common and rare species
C.common.rare	part of phylogeny similarity matrix between common (columns) and rare (rows) species
interceptVarInflation	multiplier for conditional variance prior of intercept term in the CORAL prior, compared to phylogeny-induced conditional prior

Details

The returned CORAL prior components are matrices with rows corresponding to the rare species from the spNames.rare argument.

Value

A list containing CORAL prior as matrix of means and matrix of flattened covariances

---

coralGetXDataExt *coralGetXDataExt*

Description

Constructs extended XData and XFormula from fitted Hmsc for consequent CORAL analysis

Usage

```
coralGetXDataExt(m, nf = NULL, varProp = NULL)
```

Arguments

m	fitted Hmsc-class object
nf	upper number of leading latent factors to be used in CORAL analysis per HmscRandomLevel
varProp	proportion of explanatory variance for selection of the number of leading latent factors

Details

This functions finds a point estimate of latent factors from fitted `m` `Hmsc`-class object and stacks `m$XData` and `m$XFormula` with this point estimate. Such point estimate of latent factors is selected among the posterior MCMC samples of the `m`, specifically the one with top explanatory power. The selected posterior sample is potentially truncated to a smaller number of leading latent factors, based on `nf` or `varProp` arguments.

Only one of `nf` and `varProp` can be specified. Each of these arguments can be provided as vector of length `m$nr` or scalar value that is broadcasted to such vector. By default the `nf` is set to infinity, resulting in selection of all estimated latent factors.

For example, if `Hmsc` model has `XFormula="~X1+X2"` and 2 random levels `LF1`, `LF2`, with 3 and 4 latent factors extracted correspondingly, then the extended `XDataExt="~X1+X2 + LF1.1+LF1.2+LF1.3 + LF2.1+LF2.2+LF2.3+LF2.4"`.

Value

A named list of extended `XDataExt` dataframe and corresponding extended `XFormulaExt`

`coralPlotBeta` *coralPlotBeta*

Description

Plots summary of Beta coefficients fitted with CORAL approach

Usage

```
coralPlotBeta(
  mu,
  V,
  phyloTree,
  spNames.common,
  plotColumns = c(1:ncol(mu)),
  quantile.support = c(0.05, 0.95),
  plotTree = 0.3,
  seed = NULL,
  col.common = c("blue", "red"),
  alpha.common = 0.5,
  jitter.common = 0.45,
  cex.common = 0.5,
  pch.common = 16,
  col.rare = c("cyan", "pink"),
  alpha.rare = 0.5,
  jitter.rare = 0.2,
  cex.rare = 0.2,
  pch.rare = 16,
  showCovNames = TRUE
)
```

Arguments

mu	CORAL-like matrix of means
V	CORAL-like matrix of flattened variances
phyloTree	phylogenetic tree covering all species
spNames.common	species names designated as common in CORAL analysis
plotColumns	vector of covariates' indices to be included to the plot
quantile.support	what lower and upper tails of species coefficients to visualize
plotTree	proportion of the plot canvas to use for the phylogenetic tree, 0 for no tree
seed	random seed used for jittering
col.common	a vector of two color strings indicating lower and upper tails among common species
alpha.common	transparency for common species
jitter.common	jitter amount for common species
cex.common	size of points for common species
pch.common	symbol type for common species
col.rare	a vector of two color strings indicating lower and upper tails among rare species
alpha.rare	transparency for rare species
jitter.rare	jitter amount for rare species
cex.rare	size of points for rare species
pch.rare	symbol type for rare species
showCovNames	whether to display covariate names in the plot

---

coralPredict *coralPredict*

Description

Makes predictions from CORAL models

Usage

```
coralPredict(XData, XFormula, mu, V, prob = TRUE)
```

Arguments

XData	dataframe of covariates for the sampling units to be predicted
XFormula	a formula-class object for fixed effects
mu	matrix of CORAL prior or posterior means
V	matrix of CORAL prior or posterior flattened variances
prob	whether to return predictions as probabilities (TRUE) or latent linear predictor (FALSE)

Value

Matrix of CORAL predictions with predicted sampling units in the rows and species in the columns

---

`coralPreprocess` *coralPreprocess*

Description

Splits typical Hmsc data into common and rare partitions for CORAL analysis

Usage

```
coralPreprocess(
  Y,
  spNames.common,
  TrData = NULL,
  phyloTree = NULL,
  Taxa = NULL,
  TaxaFormula = NULL
)
```

Arguments

`Y`	community matrix of both common and rare species
`spNames.common`	vector of species that are considered common in CORAL analysis
`TrData`	trait matrix for both common and rare species
`phyloTree`	phylogeny tree covering both common and rare species
`Taxa`	dataframe with
`TaxaFormula`	formula for calculating phylogeny from taxonomy, as in `as.phylo.formula(...)`

Details

This functions implies that all column names of `Y` that are not listed in `spNames.common` argument are considered rare species in the context of CORAL analysis.

Exactly one argument of phyloTree and Taxa must be specified.

Value

A named list containing `Y.xyz` and `TrData.xyz` parts for common and rare partitions, phylogeny tree `phyloTree.common` for common part and phylogeny similarity matrix `C.common.rare` between common and rare species

coralSplitToBatches *coralSplitToBatches*

Description

Splits CORAL data objects that contain all rare species into smaller batches for distributed computation

Usage

```
coralSplitToBatches(Y, TrData, C.common.rare, batchN = 1)
```

Arguments

Y	community matrix of rare species
TrData	trait matrix (dataframe) for rare species
C.common.rare	phylogeny similarity matrix between common and rare species
batchN	target number of batches

Value

A named list of three elements Y, TrData and C.common.rare, each of which is a list of length batchN with splits of the corresponding input argument

coralTrain *coralTrain*

Description

Trains CORAL models

Usage

```
coralTrain(Y, XData, XFormula, prior.mu, prior.V, transient, samples, thin)
```

Arguments

Y	covariates data.frame
XData	a data frame of measured covariates for fixed effects, same as in the backbone Hmsc(...) constructor
XFormula	a formula-class object for fixed effects
prior.mu	matrix of CORAL prior means
prior.V	matrix of CORAL prior variances
transient	number of transient MCMC steps
samples	number of posterior MCMC samples
thin	thinning interval for MCMC samples

Details

Arguments `transient`, `samples` and `thin` are passed to the `MCMCprobit(...)` call sequentially made for for each species.

Value

list with means and covariance matrices of CORAL posteriors

`createPartition` *createPartition*

Description

Constructs a partition vector given the number of folds and column of study design

Usage

```
createPartition(hM, nfolds = 10, column = NULL)
```

Arguments

`hM`	a fitted `Hmsc` model object
`nfolds`	number of cross-validation folds
`column`	name or index of the column in the `studyDesign` matrix, corresponding to the level for which units are splitted into folds

Value

a vector describing the fold of each sampling unit

Examples

```
# Create 3 folds for a HMSC object
partition = createPartition(TD$m, nfolds = 3)
```

evaluateModelFit *evaluateModelFit*

Description

Computes measures of model fit for a Hmsc model

Usage

```
evaluateModelFit(hM, predY)
```

Arguments

hM	a fitted Hmsc model object
predY	array of predictions, typically posterior sample

Details

All measures of model fit are based on comparing the posterior predictive distribution (predY)) to the observed values (hM$Y). The predicted distribution is first summarized to a single matrix of predicted values by taking the posterior mean (for normal and probit models) or posterior median (for Poisson models). All measures of model fit are given as vectors with one value for each species.

The kinds of measures of model fit depend on the type of response variable. For all types of response variables, root-mean-square error (RMSE) between predicted and observed values is computed. For normal models, R2 is computed as squared pearson correlation between observed and predicted values, times the sign of the correlation. For probit models, Tjur R2 and AUC are computed. For Poisson models, a pseudo-R2 is computed as squared spearman correlation between observed and predicted values, times the sign of the correlation (SR2). For Poisson models, the observed and predicted data are also truncated to occurrences (presence-absences), for which the same measures are given as for the probit models (O.RMSE, O.AUC and O.TjurR2). For Poisson models, the observed and predicted data are also subsetted to conditional on presence, for which the root-mean-square error and pseudo-R2 based on squared spearman correlation are computed (C.RMSE, C.SR2).

The measures O.RMSE, O.AUC, O.TjurR2, C.RMSE and C.SR2 can be computed only if the option expected=FALSE has been used when making the predictions

If the model includes a mixture of response variable types, the resulting measures of model fit contain NA's for those response variables for which they cannot be computed.

Value

a list of measures of model fit

Examples

```
# Evaluate model fit
preds = computePredictedValues(TD$m)
MF = evaluateModelFit(hM=TD$m, predY=preds)
```

```
# Evaluate model performance based on cross validation: this will be slow
## Not run:
partition = createPartition(TD$m, nfolds = 2)
predsCV1 = computePredictedValues(TD$m, partition=partition)
MF = evaluateModelFit(hM=TD$m, predY=predsCV1)

## End(Not run)
```

---

getPostEstimate *getPostEstimate*

Description

Calculates mean, support and other posterior quantities for a specified model parameter

Usage

```
getPostEstimate(
  hM,
  parName,
  r = 1,
  x = NULL,
  q = c(),
  chainIndex = 1:length(hM$postList),
  start = 1,
  thin = 1
)
```

Arguments

hM	a fitted Hmsc model object
parName	name of the parameter to be summarized. Can take value of model's baseline parameters, "Omega" or "OmegaCor".
r	the random level for which to calculate the parameter. Has effect only for Eta, Lambda, Omega and OmegaCor.
x	values of covariates for covariate dependent omega
q	vector of quantiles to calculate.
chainIndex	which posterior chains to use for summarization (defaults to all)
start	index of first MCMC sample included
thin	thinning interval of posterior distribution

Value

A named list of posterior quantities.

Examples

```
# Get posterior mean and support for species' responses to environmental covariates
postBeta = getPostEstimate(TD$m, parName='Beta')

# Get posterior mean and support for species' responses to latent factors for the first random level
postLambda = getPostEstimate(TD$m, parName='Lambda', r=1)
```

Hmsc *Hmsc*

Description

Creates an Hmsc-class object

Usage

```
Hmsc(
  Y,
  XFormula = ~.,
  XData = NULL,
  X = NULL,
  XScale = TRUE,
  XSelect = NULL,
  XRRRData = NULL,
  XRRRFormula = ~. - 1,
  XRRR = NULL,
  ncRRR = 2,
  XRRRScale = TRUE,
  YScale = FALSE,
  Loff = NULL,
  studyDesign = NULL,
  ranLevels = NULL,
  ranLevelsUsed = names(ranLevels),
  TrFormula = NULL,
  TrData = NULL,
  Tr = NULL,
  TrScale = TRUE,
  phyloTree = NULL,
  C = NULL,
  distr = "normal",
  truncateNumberOfFactors = TRUE
)
```

Arguments

Y	a matrix of species occurences or abundances

XFormula	a formula-class object for fixed effects (linear regression)
XData	a data frame or list of data frames of measured covariates for fixed effects with formula-based specification
X	a matrix of measured covariates for fixed effects with direct specification
XScale	a boolean flag indicating whether to scale covariates for the fixed effects
XSelect	a list describing how variable selection is to be applied
XRRRData	a data frame of covariates for reduced-rank regression
XRRRFormula	formula for reduced-rank regression
XRRR	a matrix of covariates for reduced-rank regression
ncRRR	number of covariates (linear combinations) for reduced-rank regression
XRRRScale	a boolean flag indicating whether to scale covariates for reduced-rank regression
YScale	a boolean flag whether to scale responses for which normal distribution is assumed
Loff	an offset additive term that is added to the linear predictor, same shape as Y
studyDesign	a data frame of correspondence between sampling units and units on different levels of latent factors
ranLevels	a named list of HmscRandomLevel-class objects, specifying the structure and data for random levels
ranLevelsUsed	a vector with names of levels of latent factors that are used in the analysis
TrFormula	a formula-class object for regression dependence of β_{kj} coefficients on species traits
TrData	a data frame of measured species traits for formula-based specification
Tr	a matrix of measured traits for direct specification
TrScale	a boolean flag whether to scale values of species traits
phyloTree	a phylogenetic tree (object of class phylo or corPhyl) for species in Y
C	a phylogenic correlation matrix for species in Y
distr	a string shortcut or $n_s \times 2$ matrix specifying the observation models
truncateNumberOfFactors	logical, reduces the maximal number of latent factor to be at most the number of species

Details

Matrix Y may contain missing values, but it is not recommended to add a species/sampling unit with fully missing data, since those do not bring any new additional information.

Only one of XFormula-XData and X arguments can be specified. Similar requirement applies to TrFormula-TrData and Tr. It is recommended to use the specification with formula, since that information enables additional features for postprocessing of the fitted model.

Besides specifying identical covariates for all species with XData being a dataframe, it is possible to provide species-specific sets of covariates by passing a list of dataframes of similar shape and structure.

As default, scaling is applied for X and Tr matrices, but not for Y matrix. If the X and/or Tr matrices are scaled, the estimated parameters are back-transformed so that the estimated parameters correspond to the original X and Tr matrices, not the scaled ones. In contrast, if Y is scaled, the estimated parameters are not back-transformed because doing so is not possible for all model parameters. Thus, the estimated parameters correspond to the scaled Y matrix, not the original one. If the Y matrix is scaled, the predictions generated by predict are back-transformed, so that the predicted Y matrices are directly comparable to the original Y matrix. If default priors are assumed, it is recommended that all matrices (X, Tr and Y) are scaled.

The object XSelect is a list. Each object of the list Xsel = XSelect[[i]] is a named list with objects Xsel$covGroup, Xsel$spGroup and Xsel$q. The parameter covGroup is a vector containing the columns of the matrix X for which variable selection is applied. The parameter spGroup is a vector of length equal to the number of species ns, with values 1,...,ng, where ng is the number of groups of species for which variable selection is applied simultanously. The parameter q is a vector of length ng, containing the prior probabilities by which the variables are to be included. For example, choosing covGroup = c(2,3), spGroup = rep(1,ns) and q=0.1 either includes or excludes both of the covariates 2 and 3 simultaneously for all species. For another example, choosing covGroup = c(2,3), spGroup = 1:ns and q=rep(0.1,ns) either includes or excludes both of the covariates 2 and 3 separately for each species.

The included random levels are specified by the ranLevels and ranLevelsUsed arguments. The correspondence between units of each random level and rows of Y must be specified by a column of studyDesign, which corresponds to the name of a list item in ranLevels. It is possible to provide an arbitrary number of columns in studyDesign that are not listed in ranLevels. These do not affect the model formulation or fitting scheme, but can be utilized during certain functions postprocessing the results of statistical model fit.

The distr argument may be either a matrix, a string literal, or a vector of string literals. In the case of a matrix, the dimension must be $n_s \times 2$, where the first column defines the family of the observation model and the second argument defines the dispersion property. The elements of the first column must take values 1-normal, 2-probit and 3-Poisson with log link function. The second argument stands for the dispersion parameter being fixed (0) or estimated (1). The default fixed values of the dispersion parameters are 1 for normal and probit, and 0.01 for Poisson (implemented as a limiting case of lognormally-overdispersed Poisson). Alternatively, a string literal shortcut can be given as a value to the distr argument, simultaniously specifying similar class of observation models for all species. The available shortcuts are "normal", "probit", "poisson", "lognormal poisson". If distr is a vector of string literals, each element corresponds to one species, should be either "normal", "probit", "poisson", "lognormal poisson", and these can be abbreviated as long as they are unique strings. The matrix argument and the vector of string literals allows specifying different observation models for different species.

The offset term Loff enables to add a pre-defined quantity to the linear predictor. This can be used to account for variable sampling effort across sampling units and species.

By default this constructor assigns default priors to the latent factors. Those priors are designed to be reasonably flat assuming that the covariates, species traits and normally distributed responses are scaled. In case when other priors needed to be specified, a call of setPriors.Hmsc methods should be made, where the particular priors may be specified.

Value

An object of Hmsc class without any posterior samples.

See Also

HmscRandomLevel, sampleMcmc, setPriors.Hmsc

Examples

```
# Creating a Hmsc object without phylogeny, trait data or random levels
m = Hmsc(Y=TD$Y, XData=TD$X, XFormula=~x1+x2)

# Creating a Hmsc object with phylogeny and traits
m = Hmsc(Y=TD$Y, XData=TD$X, XFormula=~x1+x2,
TrData=TD$Tr, TrFormula=~T1+T2, phyloTree=TD$phylo)

# Creating a Hmsc object with 2 nested random levels (50 sampling units in 20 plots)
studyDesign = data.frame(sample = as.factor(1:50), plot = as.factor(sample(1:20,50,replace=TRUE)))
rL1 = HmscRandomLevel(units=levels(TD$studyDesign$plot))
rL2 = HmscRandomLevel(units=levels(TD$studyDesign$sample))
m = Hmsc(Y=TD$Y, XData=TD$X, XFormula=~x1+x2,
studyDesign=studyDesign,ranLevels=list("sample"=rL1,"plot"=rL2))
```

HmscRandomLevel *Create an* Hmsc *random level*

Description

Specifies the structure of a random factor, including whether the random factor is assumed to be spatially explicit or not, the spatial coordinates and the potential structure of covariate-dependent random factors.

Usage

```
HmscRandomLevel(
  sData = NULL,
  sMethod = "Full",
  distMat = NULL,
  xData = NULL,
  units = NULL,
  N = NULL,
  nNeighbours = 10,
  sKnot = NULL,
  longlat = FALSE
)
```

Arguments

sData	a matrix or a dataframe containing spatial or temporal coordinates of units of the random level, or a similar SpatialPoints structure of the **sp** package. If spatial coordinates are unprojected longitude and latitude, great circle distances will be

calculated internally. All spatial locations should be unique. If you have several observations in the same point, they should be identified by the random levels.

`sMethod` a string specifying which spatial method to be used. Possible values are `"Full"`, `"GPP"` and `"NNGP"`

`distMat` a distance matrix containing the distances between units of the random level, with unit names as rownames, or a `dist` structure with location Labels. `distMat` cannot be used with `"GPP"` spatial model.

`xData` a dataframe containing the covariates measured at the units of the random level for covariate-dependent associations

`units` a vector, specifying the names of the units of a non-structured level

`N` number of unique units on this level

`nNeighbours` a scalar specifying the number of neighbours to be used in case the spatial method is set to `NNGP`. Only positive values smaller than the total number of plots are allowed.

`sKnot` a dataframe containing the knot locations to be used for the Gaussian predictive process if `sMethod` is set to `"GPP"`. Suitable data can be produced with `constructKnots`. The knot locations shall not duplicate `sData`.

`longlat` Interpret coordinate data `sData` as longitude and latitude in decimal degrees. If this is `TRUE`, great circle distances will be used instead of Euclidean distances.

Details

Only one of `sData`, `distMat`, `xData`, `units` and `N` arguments can be provided.

As a good practice, we recommend to specify all available units for a random level, even if some of those are not used for training the model.

Value

a `HmscRandomLevel`-class object that can be used for `Hmsc`-class object construction

See Also

`setPriors.Hmsc` to change the default priors of an existing `HmscRandomLevel` object.

Examples

```
# Setting a random level with 50 units
rL = HmscRandomLevel(units=TD$studyDesign$sample)

# Setting a spatial random level
rL = HmscRandomLevel(sData=TD$xycoords)

# Setting a covariate-dependent random level.
rL = HmscRandomLevel(xData=data.frame(x1=rep(1,length(TD$X$x1)),x2=TD$X$x2))
```

importPosteriorFromHPC *importPosteriorFromHPC*

Description

Integrates Hmsc model with a list of chains containing posterior MCMC samples generated with Hmsc-HPC

Usage

```
importPosteriorFromHPC(
  m,
  postList,
  nSamples,
  thin,
  transient,
  adaptNf = rep(transient, m$nr),
  verbose = 0,
  alignPost = TRUE
)
```

Arguments

m	a Hmsc model object for which the posterior will be imported
postList	list of MCMC chains, where each chain is a list of posterior samples.
nSamples	number of posterior MCMC samples per chain used in Hmsc-HPC
thin	thinning interval used in Hmsc-HPC
transient	number of transient MCMC steps used in Hmsc-HPC
adaptNf	number of MCMC steps for adjustung number of factors used in Hmsc-HPC
verbose	level of verbose to fake
alignPost	boolean flag indicating whether the posterior of each chains should be aligned

Details

Naturally, the model object m must be compatible with the posterior samples imported from Hmsc-HPC. If m already contains posterior samples, these will be overwritten.

Parameters nSamples, thin, transient, adaptNf and especially verbose are required to guarantee that the resulted Hmsc object works interchangeably with post-processing functions in the Hmsc package as if the sampling was done in the Hmsc R package. These values shall mirror the corresponding values that would be used in the sampleMcmc(...) call.

Parameter alignPost has an identical effect to its counterpart from sampleMcmc(...).

Value

a fitted Hmsc model object

Examples

```
if(FALSE){
# If all chains were run with a single Hmsc-HPC call
post_file_path = ... # path to the file where the posterior samples are stored
importFromHPC = from_json(readRDS(file = post_file_path)[[1]])
postList = importFromHPC[1:nChains]
cat(sprintf("fitting time %.1f sec\n", importFromHPC[[nChains+1]]))
fm = importPosteriorFromHPC(m, postList, nSamples, thin, transient)

# If Hmsc-HPC calls were configured to run one chain per call
post_file_path = ... # directory with Hmsc-HPC export files stored as post_chainXX_file.rds
chainList = vector("list", nChains)
for(cInd in 1:nChains){
  chain_file_path = file.path(post_file_path, sprintf("post_chain%.2d_file.rds", cInd-1))
   chainList[[cInd]] = from_json(readRDS(file = chain_file_path)[[1]])[[1]]
}
fm = importPosteriorFromHPC(m, postList, nSamples, thin, transient)
}
```

---

plotBeta *plotBeta*

Description

Plots heatmaps of parameter estimates or posterior support values of species' environmental responses, i.e. how species in `Y` responds to covariates in `X`

Usage

```
plotBeta(
  hM,
  post,
  param = "Support",
  plotTree = FALSE,
  SpeciesOrder = "Original",
  SpVector = NULL,
  covOrder = "Original",
  covVector = NULL,
  spNamesNumbers = c(TRUE, TRUE),
  covNamesNumbers = c(TRUE, TRUE),
  supportLevel = 0.9,
  split = 0.3,
  cex = c(0.7, 0.7, 0.8),
```

```
  colors = colorRampPalette(c("blue", "white", "red")),
  colorLevels = NULL,
  mar = NULL,
  marTree = c(6, 0, 2, 0),
  mgp = c(3, 2, 0),
  main = NULL,
  bigplot = NULL,
  smallplot = NULL,
  newplot = TRUE
)
```

Arguments

hM	a fitted Hmsc model object
post	posterior summary of Beta parameters obtained from getPostEstimate
param	controls which parameter is plotted, current options include "Mean" for posterior mean estimate, "Support" for the level of statistical support measured by posterior probability for a positive or negative response, and "Sign" to indicate whether the response is positive, negative, or neither of these given the chosen supportLevel
plotTree	logical. Whether species' environmental responses is to be mapped onto the phylogeny used in model fitting
SpeciesOrder	controls the ordering of species, current options are "Original", "Tree", and "Vector". If SpeciesOrder = "Vector", an ordering vector must be provided (see SpVector). If plotTree = TRUE, SpeciesOrder is ignored
SpVector	controls the ordering of species if SpeciesOrder = "Vector". If a subset of species are listed, only those will be plotted. For alphabetic ordering, try match(1:hM$ns, as.numeric(as.factor(colnames(hM$Y))))
covOrder	controls the ordering of covariates, current options are "Original" and "Vector". If covOrder = "Vector", an ordering vector must be provided (see covVector)
covVector	controls the ordering of covariates if covOrder = "Vector". If a subset of covariates are listed, only those will be plotted
spNamesNumbers	logical of length 2, where first entry controls whether species names are added to axes, and second entry controls whether species numbers are added
covNamesNumbers	logical of length 2, where first entry controls whether covariate names are added to axes, and second entry controls whether covariate numbers are added
supportLevel	controls threshold posterior support for plotting
split	if plotTree = TRUE, controls the division of the plotting window between the tree and the heatmap.
cex	controls character expansion (font size). Three values, controlling covariate names, species names, and color legend axis labels
colors	controls the colors of the heatmap, default value colorRampPalette(c("blue","white","red"))
colorLevels	number of color levels used in the heatmap

mar	plotting margins
marTree	plotting margins for phylogenetic tree
mgp	can be used to set the location of the scale bar
main	main title for the plot.
bigplot	argument passed to image.plot, designates position of the heatmap
smallplot	argument passed to image.plot, designates position of the colorbar
newplot	set to false if the plot will be part of multi-panel plot initialized with par(mfrow)

Examples

```
# Plot posterior support values of species' environmental responses
betaPost=getPostEstimate(TD$m, "Beta")
plotBeta(TD$m, post=betaPost, param="Support")

# Plot parameter estimates of species' environmental responses together with the phylogenetic tree
betaPost=getPostEstimate(TD$m, "Beta")
plotBeta(TD$m, post=betaPost, param="Mean", plotTree=TRUE)
```

---

plotGamma *plotGamma*

Description

Plots heatmaps of parameter estimates or posterior support values of trait effects on species' environmental responses, i.e. how environmental responses in Beta responds to covariates in X

Usage

```
plotGamma(
  hM,
  post,
  param = "Support",
  trOrder = "Original",
  trVector = NULL,
  covOrder = "Original",
  covVector = NULL,
  trNamesNumbers = c(TRUE, TRUE),
  covNamesNumbers = c(TRUE, TRUE),
  supportLevel = 0.9,
  main = NULL,
  cex = c(0.8, 0.8, 0.8),
  colors = colorRampPalette(c("blue", "white", "red")),
  colorLevels = NULL,
  mar = c(6, 9, 2, 0),
  bigplot = NULL,
```

```
  smallplot = NULL,
  newplot = TRUE
)
```

Arguments

hM	a fitted Hmsc model object
post	posterior summary of Gamma parameters obtained from getPostEstimate
param	controls which parameter is plotted, current options include "Mean" for posterior mean estimate, "Support" for the level of statistical support measured by posterior probability for a positive or negative response, and "Sign" to indicate whether the response is positive, negative, or neither of these given the chosen supportLevel
trOrder	controls the ordering of traits, current options are "Original", and "Vector". If trOrder = "Vector", an ordering vector must be provided (see trVector)
trVector	controls the ordering of traits if trOrder = "Vector". If a subset of traits are listed, only those will be plotted
covOrder	controls the ordering of covariates, current options are "Original" and "Vector". If covOrder = "Vector", an ordering vector must be provided (see covVector)
covVector	controls the ordering of covariates if covOrder = "Vector". If a subset of covariates are listed, only those will be plotted
trNamesNumbers	logical of length 2, where first entry controls whether trait names are added to axes, and second entry controls whether traits numbers are added
covNamesNumbers	logical of length 2, where first entry controls whether covariate names are added to axes, and second entry controls whether covariate numbers are added
supportLevel	controls threshold posterior support for plotting
main	main title for the plot
cex	controls character expansion (font size). Three values, controlling covariate names, trait names, and color legend axis labels
colors	controls the colors of the heatmap, default value colorRampPalette(c("blue","white","red"))
colorLevels	number of color levels used in the heatmap
mar	plotting margins
bigplot	argument passed to image.plot, designates position of the heatmap
smallplot	argument passed to image.plot, designates position of the colorbar
newplot	set to false if the plot will be part of multi-panel plot

Examples

```
# Plot posterior support values of trait effects on environmental responses
gammaPost=getPostEstimate(TD$m, "Gamma")
plotGamma(TD$m, post=gammaPost, param="Support")

# Plot parameter estimates of trait effects on environmental responses
```

```
gammaPost=getPostEstimate(TD$m, "Gamma")
plotGamma(TD$m, post=gammaPost, param="Mean")
```

---

plotGradient *plotGradient*

Description

Plots an environmental gradient over one of the variables included in XData

Usage

```
plotGradient(
  hM,
  Gradient,
  predY,
  measure,
  xlabel = NULL,
  ylabel = NULL,
  index = 1,
  q = c(0.025, 0.5, 0.975),
  cicol = rgb(0, 0, 1, alpha = 0.5),
  pointcol = "lightgrey",
  pointsize = 1,
  showData = FALSE,
  jigger = 0,
  yshow = NA,
  showPosteriorSupport = TRUE,
  main,
  ...
)
```

Arguments

hM	a fitted Hmsc model object
Gradient	an object returned by constructGradient
predY	an object returned by applying the function predict to Gradient
measure	whether to plot species richness ("S"), an individual species ("Y") or community-weighted mean trait values ("T")
xlabel	label for x-axis
ylabel	label for y-axis
index	which species or trait to plot
q	quantiles of the credibility interval plotted
cicol	colour with which the credibility interval is plotted

pointcol	colour with which the data points are plotted
pointsize	size in which the data points are plotted
showData	whether raw data are plotted as well
jigger	the amount by which the raw data are to be jiggered in x-direction (for factors) or y-direction (for continuous covariates)
yshow	scale y-axis so that these values are also visible. This can used to scale y-axis so that it includes 0 and the expected maximum values.
showPosteriorSupport	add margin text on the posterior support of predicted change from gradient minimum to maximum for continuous gradients.
main	main title for the plot.
...	additional arguments for plot

Details

For measure="Y", index selects which species to plot from hM$spNames. For measure="T", index selects which trait to plot from hM$trNames. With measure="S" the row sum of pred is plotted, and thus the interpretation of "species richness" holds only for probit models. For Poisson models "S" shows the total count, whereas for normal models it shows the summed response. For measure="T", in probit model the weighting is over species occurrences, whereas in count models it is over individuals. In normal models, the weights are exp-transformed predictions to avoid negative weights

Value

For the case of a continuous covariate, returns the posterior probability that the plotted variable is greater for the last sampling unit of the gradient than for the first sampling unit of the gradient. For the case of a factor, returns the plot object.

See Also

constructGradient, predict

Examples

```
# Plot response of species 2 over the gradient of environmental variable x1
Gradient = constructGradient(TD$m, focalVariable="x1")
predY = predict(TD$m, Gradient=Gradient)
plotGradient(TD$m, Gradient, pred=predY, measure="Y", index = 2, showData = TRUE, jigger = 0.05)
# Plot modelled species richness over the gradient of environmental variable x1
Gradient = constructGradient(TD$m, focalVariable="x1")
predY = predict(TD$m, Gradient=Gradient)
plotGradient(TD$m, Gradient, pred=predY, measure="S")
```

plotVariancePartitioning *plotVariancePartitioning*

Description

Plots the results of variance partitioning of a Hmsc model produced by computeVariancePartitioning as a barplot

Usage

```
plotVariancePartitioning(
  hM,
  VP,
  cols = NULL,
  main = "Variance Partitioning",
  ...
)
```

Arguments

hM	a fitted Hmsc model object
VP	a Hmsc variance partitioning object produced by computeVariancePartitioning
cols	colors of the barplot
main	main title for the plot
...	additional parameters passed to the barplot function

Examples

```
# Plot how the explained variance of a previously fitted model is partitioned
VP = computeVariancePartitioning(TD$m)
plotVariancePartitioning(TD$m, VP)
```

poolMcmcChains *poolMcmcChains*

Description

Combines a list of single or several MCMC chains into a single chain

Usage

```
poolMcmcChains(postList, chainIndex = 1:length(postList), start = 1, thin = 1)
```

Arguments

postList	list of posterior chains
chainIndex	index of chains to be included
start	index of first MCMC sample included
thin	thinning between included MCMC samples

Value

a list with combined MCMC samples

Examples

```
# Combine the posteriors from all chains in a Hmsc object
postList = TD$m$postList
pooledPost = poolMcmcChains(postList)
```

---

predict.Hmsc *predict*

Description

Calculates predicted values from a fitted Hmsc model.

Usage

```
## S3 method for class 'Hmsc'
predict(
  object,
  post = poolMcmcChains(object$postList),
  Loff = NULL,
  XData = NULL,
  X = NULL,
  XRRRData = NULL,
  XRRR = NULL,
  studyDesign = object$studyDesign,
  ranLevels = object$ranLevels,
  Gradient = NULL,
  Yc = NULL,
  mcmcStep = 1,
  expected = FALSE,
  predictEtaMean = FALSE,
  predictEtaMeanField = FALSE,
  nParallel = 1,
  useSocket = TRUE,
  ...
)
```

Arguments

object	a fitted Hmsc model object
post	a list of posterior samples of the HMSC model. By default uses all samples from the pooled posterior of the hM object.
Loff	offset matrix of the same dimensions as the prediction. Added to the predictions on the linear predictor scale. Same as in the Hmsc constructor, this is used to account to pre-defined differences between prediction units.
XData	a dataframe specifying the unpreprocessed covariates for the predictions to be made. Works only if the XFormula argument was specified in the Hmsc() model constructor call. Requirements are similar to those in the Hmsc model constructor.
X	a matrix specifying the covariates for the predictions to be made. Only one of XData and X arguments may be provided.
XRRRData	a dataframe of covariates for reduced-rank regression
XRRR	a matrix of covariates for reduced-rank regression
studyDesign	a matrix, specifying the structure of the study design for the prediction. Requirements are similar to those of the Hmsc constructor. By default this argument is assigned the study design of the training data in the fitted Hmsc model.
ranLevels	a list of HmscRandomLevel objects, futher specifying the structure of random levels. Requirements are similar to those of the Hmsc constructor. Each level must cover all units, specified in the correspondingly named column of studyDesign argument. By default this argument is assigned the list of HmscRandomLevel objects specified for fitting Hmsc model.
Gradient	an object returned by constructGradient. Providing Gradient is an alternative for providing XData, studyDesign and ranLevels. Cannot be used together with Yc.
Yc	a matrix of the outcomes that are assumed to be known for conditional predictions. Cannot be used together with Gradient and use with caution for spatial models (see details).
mcmcStep	the number of extra mcmc steps used for updating the random effects
expected	boolean flag indicating whether to return the location parameter of the observation models or sample the values from those.
predictEtaMean	boolean flag indicating whether to use the estimated mean values of posterior predictive distribution for random effets corresponding for the new units.
predictEtaMeanField	boolean flag indicating whether to use draws from the mean-field of the posterior predictive distribution for random effets corresponding for the new units.
nParallel	Number of parallel processes. Parallel processing is only useful with new Yc data and extra mcmcStep.
useSocket	(logical) Use socket clusters in parallel proecessing; these are the only alternative in Windows, but in other systems this should be usually set FALSE for forking.
...	other arguments passed to functions.

Details

In case of conditional predictions (once non null `Yc` is provided) `mcmcStep`,the number of extra mcmc steps used for updating the random effects for the Eta parameters, starting from the samples of the fitted Hmsc model in order to account for the conditional infromation provided in the Yc argument. The higher this number is, the more the obtained updated samples are unaffected by the posterior estimates of latent factors in the model fitted to the training data and more resembles the true conditional posterior. However, the elapsed time for conditional prediction grows approximately linearly as this parameter increases. The exact number for sufficient is problem-dependent and should be assessed by e.g. gradually increasing this parameter till the stationarity of the produced predictions.

Note that the currently implemented conditional predictions behavior is well-formulated once the sampling units in `Yc` are either the same as in the originally fitted model `Y`, or are considered independent of those. Thus, if seeking such conditional predictions at new loacations for a spatial `Hmsc` model, or the case once some units of `HmscRandomLevel` belong both to the training and predictions sets of sampling units, the current implementations is not guaranteed to correctly operate in intended way.

Value

A list of length `length(post)`, each element of which contains a sample from the posterior predictive distribution (given the sample of the Hmsc model parameters in the corresponding element of the `post` argument)

See Also

`predictLatentFactor`

`predictLatentFactor` *predictLatentFactor*

Description

Draws samples from the conditional predictive distribution of latent factors

Usage

```
predictLatentFactor(
  unitsPred,
  units,
  postEta,
  postAlpha,
  rL,
  predictMean = FALSE,
  predictMeanField = FALSE
)
```

Arguments

unitsPred	a factor vector with random level units for which predictions are to be made
units	a factor vector with random level units that are conditioned on
postEta	a list containing samples of random factors at conditioned units
postAlpha	a list containing samples of range (lengthscale) parameters for latent factors
rL	a HmscRandomLevel-class object that describes the random level structure
predictMean	a boolean flag indicating whether to return the mean of the predictive Gaussian process distribution
predictMeanField	a boolean flag indicating whether to return the samples from the mean-field distribution of the predictive Gaussian process distribution

Details

Length of units vector and number of rows in postEta matrix shall be equal. The method assumes that the i-th row of postEta correspond to i-th element of units.

This method uses only the coordinates rL$s field of the rL$s argument. This field shall be a matrix with rownames covering the union of unitsPred and units factors. Alternatively, it can use distance matrix rL$distMat which is a symmetric square matrix with similar row names as the coordinate data (except for the GPP models that only can use coordinates).

In case of spatial random level, the computational complexity of the generic method scales cubically as the number of unobserved units to be predicted. Both predictMean=TRUE and predictMeanField=TRUE options decrease the asymptotic complexity to linear. The predictMeanField=TRUE option also preserves the uncertainty in marginal distribution of predicted latent factors, but neglects the inter-dependece between them.

Value

a list of length length(postEta) containing samples of random factors at unitsPred from their predictive distribution conditional on the values at units

---

prepareGradient *prepareGradient*

Description

prepares a user-made environmental and/or spatial gradient to be used for prediction

Usage

```
prepareGradient(hM, XDataNew, sDataNew)
```

Arguments

hM	a fitted Hmsc model object.
XDataNew	a dataframe of the new XData.
sDataNew	a named list of the new sData, where the name gives the spatial random level.

Details

The dataframe XDataNew is the for output as for input. The main purpose of this function is to prepare the study design and random levels so that predictions can be made with the predict function. Note that the difference between constructGradient and prepareGradient is that while prepareGradient takes as input the new environmental and spatial data, constructGradient generates those data to represent a new environmental gradient.

Value

a named list with members XDataNew, studyDesignNew and rLNew

See Also

constructGradient, predict

---

sampleMcmc *sampleMCMC*

Description

Samples the posterior with block-conditional Gibbs MCMC sampler

Usage

```
sampleMcmc(
  hM,
  samples,
  transient = 0,
  thin = 1,
  initPar = NULL,
  verbose,
  adaptNf = rep(transient, hM$nr),
  nChains = 1,
  nParallel = 1,
  useSocket = TRUE,
  dataParList = NULL,
  updater = list(Gamma2 = FALSE, GammaEta = FALSE),
  fromPrior = FALSE,
  alignPost = TRUE,
  engine = "R"
)
```

Arguments

hM	a fitted Hmsc model object
samples	the number of MCMC samples to be obtained in each chain
transient	the number of MCMC steps that are executed before starting recording posterior samples
thin	the number of MCMC steps between each recording of samples from the posterior
initPar	a named list of parameter values used for initialization of MCMC states, or alternatively text "fixed effects" to use linear Maximum Likelihood model instead of randomizing from prior; the "fixed effects" can shorten the transient phase of sampling, but will initialize all chains to the same starting values
verbose	the interval between MCMC steps printed to the console (default is an interval that prints ca. 50 reports)
adaptNf	a vector of length n_r with number of MCMC steps at which the adaptation of the number of latent factors is conducted
nChains	number of independent MCMC chains to be run
nParallel	number of parallel processes by which the chains are executed.
useSocket	(logical) use socket clusters in parallel processing; in Windows this is the only option, but in other operating systems fork clusters are a better alternative, and this should be set FALSE.
dataParList	a named list with pre-computed Qg, iQg, RQg, detQg, rLPar parameters
updater	a named list, specifying which conditional updaters should be ommitted
fromPrior	whether prior (TRUE) or posterior (FALSE) is to be sampled
alignPost	boolean flag indicating whether the posterior of each chains should be aligned
engine	The toolset used in MCMC chain. Currently only "R" is implemented (this argument is for developers only: see source code).

Details

The exact number of samples to be recorded in order to get a proper estimate of the full posterior with Gibbs MCMC algorithms, as well as the required thinning and cut-off of transient is very problem-specific and depends both on the model structure and the data itself. Therefore, in general it is very challenging to a priori provide an informed recommendation on what values should be used for a particular problem. A common recommended strategy involves executing the posterior sampling with MCMC with some guess of the values for these arguments, checking the properties of the obtained samples (primarily potential scale reduction factor and effective sample size), and adjusting the guess accordingly.

The value of 1 for thin argument means that at each MCMC step after the transient a sample is recorded.

Typically, the value of nParallel equal to nChains leads to most efficient usage of available parallelization capacities. However, this may be not the case if R is configured with multi-tread linear algebra libraries. For debug and test purposes, the nParallel should be set to 1, since only in this case a details of the potentially encountered errors would be available.

The dataParList argument may be handy for large problems that needs to be refitted multiple times, e.g. with different prior values. In that case, the data parameters that are precomputed for the Hmsc sampling scheme may require an undesirably lot of storage space if they are saved for each of the model. Instead, they could be computed only once and then directly reused, therefore reducing the storing redundancy.

Some of the available conditional updaters partially duplicate each other. In certain cases, the usage of all of them may lead to suboptimal performance, compared to some subset of those. Then, it is possible to manually disable some of them, by adding a $UPDATER_NAME=FALSE pair to the updater argument. Another usage of this argument involves cases when some of the model parameters are known and have to be fixed. However, such tweaks of the sampling scheme should be done with caution, as if compromized they would lead to erroneuos results.

Value

An Hmsc-class object with chains of posterior samples added to the postList field

See Also

Hmsc

Examples

```
## you need 1000 or more samples, but that will take too long
## in an example
m = sampleMcmc(TD$m, samples=10)

## Not run:
## Record 1000 posterior samples while skipping 1 MCMC step between samples
## from 2 chains after discarding the first 500 MCMC steps
m = sampleMcmc(TD$m, samples=1000, transient=500, thin=2, nChains=2, nParallel=1)

## End(Not run)
```

---

samplePrior *samplePrior*

Description

Samples the parameter vector from prior

Usage

```
samplePrior(hM, dataParList = NULL)
```

Arguments

hM	a fitted Hmsc model object
dataParList	list of data parameters (see computeDataParameters)

Value

A named list containing the Hmsc model parameters

`setPriors` *setPriors*

Description

Sets or resets priors to objects

Usage

```
setPriors(...)
```

Arguments

...	Hmsc or HmscRandolLevel object and other arguments.

Value

Object of same type as first input

See Also

setPriors.Hmsc, setPriors.HmscRandomLevel

Examples

```
# Set priors for random level so that there is minimum of 2 latent factors and maximum of 3
rL1 = HmscRandomLevel(units=TD$studyDesign$plot)
rL1 = setPriors(rL1, nfMax=3, nfMin=2)

# Set shrinkage parameters for priors of random level
rL1 = HmscRandomLevel(units=TD$studyDesign$plot)
rL1 = setPriors(rL1, a1=10, a2=10, b1=1, b2=1)
```

setPriors.Hmsc *setPriors.Hmsc*

Description

Sets or resets priors to the Hmsc object

Usage

```
## S3 method for class 'Hmsc'
setPriors(
  hM,
  V0 = NULL,
  f0 = NULL,
  mGamma = NULL,
  UGamma = NULL,
  aSigma = NULL,
  bSigma = NULL,
  nuRRR = NULL,
  a1RRR = NULL,
  b1RRR = NULL,
  a2RRR = NULL,
  b2RRR = NULL,
  rhopw = NULL,
  setDefault = FALSE,
  ...
)
```

Arguments

hM	a fitted Hmsc model object
V0	scale matrix in the Wishart prior distribution for the V matrix
f0	number of degrees of freedom in the Wishart prior distribution for the V matrix
mGamma	mean for the prior multivariate Gaussian distribution for Gamma parameters
UGamma	covariance matrix for the prior multivariate Gaussian distribution for Gamma parameters
aSigma	shape parameter for the prior gamma distribution for the variance parameter, only for normal & lognormal Poisson models
bSigma	rate parameter for the prior gamma distribution for the variance parameter, only for normal & lognormal Poisson models
nuRRR, a1RRR, b1RRR, a2RRR, b2RRR	parameters of the multiplicative gamma process shrinking prior for reduced rank regression
rhopw	discrete grid prior for phylogenetic signal, should be a matrix of 2 columns
setDefault	logical indicating whether default priors should be used
...	other parameters passed to the function.

Value

Modified Hmsc object

`setPriors.HmscRandomLevel` *setPriors.HmscRandomLevel*

Description

Sets or resets priors to the Hmsc object

Usage

```
## S3 method for class 'HmscRandomLevel'
setPriors(
  rL,
  nu = NULL,
  a1 = NULL,
  a2 = NULL,
  b1 = NULL,
  b2 = NULL,
  alphapw = NULL,
  nfMax = NULL,
  nfMin = NULL,
  setDefault = FALSE,
  ...
)
```

Arguments

rL	a fitted HmscRandomLevel model object
nu, a1, b1, a2, b2	parameters of the multiplicative gamma process shrinking prior
alphapw	discrete grid prior for spatial scale parameter
nfMax	maximum number of latent factors to be sampled
nfMin	minimum number of latent factors to be sampled
setDefault	logical indicating whether default priors should be used
...	other arguments (ignored)

Value

Modified HmscRandomLevel object

TD *Simulated data and a fitted Hmsc model for a small species community.*

Description

This dataset contains simulated occurence data for 4 species in 50 sampling units. The data is based on a hierarchical study design consisting of 50 sampling units in 10 georeferenced plots. Occurences of 4 species were simulated using one continuous environmental variable (x1) and spatial autocorrelation between the plots. Response of species to the environment are related to one species trait which is fully phylogenetically structured. This dataset is used for the examples and package testing. The variables are as follows:

Usage

```
TD
```

Format

A list of 12 objects

ns Number of species in the dataset

units Number of sampling units

plots Number of plots

X A 3 by 50 environmental matrix consisting of one continuous and one categorical variable. Also includes intercept column

phy A list containing the simulated phylogenetic tree for 4 species

C A 4 by 4 phylogenetic variance covariance matrix

Tr A 4 by 3 trait matrix with one phylogenetically phylogenetically structured continuous trait, one categroical trait and an intercept

xycoords simulated 2 dimensional coordinates

studyDesign Sampling unit and plot IDs

Y Simulated species occurences

m A fitted Hmsc object with 100 posterior samples

Index