Package: boral (via r-universe)

October 6, 2026

Title Bayesian Ordination and Regression AnaLysis

Version 2.0.3

Date 2025-05-1

Description Bayesian approaches for analyzing multivariate data in ecology. Estimation is performed using Markov Chain Monte Carlo (MCMC) methods via Three. JAGS types of models may be fitted: 1) With explanatory variables only, boral fits independent column Generalized Linear Models (GLMs) to each column of the response matrix; 2) With latent variables only, boral fits a purely latent variable model for model-based unconstrained ordination; 3) With explanatory and latent variables, boral fits correlated column GLMs with latent variables to account for any residual correlation between the columns of the response matrix.

License GPL-2

Depends coda

Imports abind, corpcor, fishMod, graphics, grDevices, lifecycle, MASS, mvtnorm, R2jags, reshape2, stats

Suggests mvabund (>= 4.0.1), corrplot

NeedsCompilation no

Author Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer Francis K.C. Hui <fhui28@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2025-04-29 23:10:02 UTC

RemoteUrl https://github.com/cran/boral

RemoteRef HEAD

RemoteSha 6e5473befed84b06e879586eb625580e7152c319

Contents

boral-package . . . 2
about.distributions . . . 3
about.lvs . . . 6
about.ranefs . . . 8
about.ssvs . . . 10
about.traits . . . 14
boral . . . 17
calc.condlogLik . . . 34
calc.logLik.lv0 . . . 37
calc.marglogLik . . . 41
calc.varpart . . . 45
coefsplot . . . 49
create.life . . . 51
ds.residuals . . . 59
fitted.boral . . . 61
get.dic . . . 63
get.enviro.cor . . . 64
get.hpdintervals . . . 66
get.mcmcsamples . . . 70
get.measures . . . 71
get.more.measures . . . 75
get.residual.cor . . . 79
lvsplot . . . 82
make.jagsboralmodel . . . 85
make.jagsboralnullmodel . . . 90
plot.boral . . . 95
predict.boral . . . 97
ranefsplot . . . 103
summary.boral . . . 105
tidyboral . . . 106

Index **110**

`boral-package` *Bayesian Ordination and Regression AnaLysis (boral)*

Description

Bayesian approaches for analyzing multivariate data in ecology. Estimation is performed using Markov Chain Monte Carlo (MCMC) methods via Three. JAGS types of models may be fitted: 1) With explanatory variables only, boral fits independent column Generalized Linear Models (GLMs) to each column of the response matrix; 2) With latent variables only, boral fits a purely latent variable model for model-based unconstrained ordination; 3) With explanatory and latent variables, boral fits correlated column GLMs with latent variables to account for any residual correlation between the columns of the response matrix.

Details

Package:	boral
Type:	Package
Version:	0.6
Date:	2014-12-12
License:	GPL-2

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Hui et al. (2014). Model-based approaches to unconstrained ordination. Methods in Ecology and Evolution, 6, 399-411.
- Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In Proceedings of the 3rd International Workshop on Distributed Statistical Computing. March (pp. 20-22).
- Skrondal, A., and Rabe-Hesketh, S. (2004). Generalized latent variable modeling: Multilevel, longitudinal, and structural equation models. CRC Press.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, 30, 766-779.
- Yi W. et al. (2013). `mvabund`: statistical methods for analysing multivariate abundance data. R package version 3.8.4.

Examples

```
## Please see main boral function for examples.
```

`about.distributions` *Distributions available in boral*

Description

[Stable]

This help file provides more information regarding the distributions i.e., the `family` argument, available in the boral package, to handle various responses types.

Details

A variety of families are available in boral, designed to accommodate multivariate abundance data of varying response types. Please see the `family` argument in the `boral` which lists all distributions that are currently available.

For multivariate abundance data in ecology, species counts are often overdispersed. Using a negative binomial distribution (`family = "negative.binomial"`) to model the counts usually helps to account for this overdispersion. Please note the variance for the negative binomial distribution is parameterized as $Var(y) = \mu + \phi\mu^2$, where ϕ is the dispersion parameter.

For non-negative continuous data such as biomass, the lognormal, Gamma, and tweedie distributions may be used (Foster and Bravington, 2013). For the gamma distribution, the variance is parameterized as $Var(y) = \mu/\phi$ where ϕ is the response-specific rate (henceforth referred to also as dispersion parameter).

For the tweedie distribution, a common power parameter is across all columns with this family, because there is almost always insufficient information to model response-specific power parameters. Specifically, the variance is parameterized as $Var(y) = \phi\mu^p$ where ϕ is the response-specific dispersion parameter and p is a power parameter common to all columns assumed to be tweedie, with $1 < p < 2$.

Normal responses are also implemented, just in case you encounter normal stuff in ecology (pun intended)! For the normal distribution, the variance is parameterized as $Var(y) = \phi^2$, where ϕ is the response-specific standard deviation.

The beta distribution can be used to model data between values between but *not* including 0 and 1. In principle, this would make it useful for percent cover data in ecology, if it not were for the fact that percent cover is commonly characterized by having lots of zeros (which are not permitted for beta regression). An *ad-hoc* fix to this would be to add a very small value to shift the data away from exact zeros and/or ones. This is however heuristic, and pulls the model towards producing conservative results (see Smithson and Verkuilen, 2006, for a detailed discussion on beta regression, and Korhonen et al., 2007, for an example of an application to forest canopy cover data). Note the parameterization of the beta distribution used here is directly in terms of the mean μ and the dispersion parameter ϕ (more commonly know as the "sample size"). In terms of the two shape parameters, if we denote the two shape parameters as the vector (a, b), his is equivalent to $a = \mu\phi$ and $b = (1 - \mu)\phi$.

For ordinal response columns, cumulative probit regression is used (Agresti, 2010). boral assumes all ordinal columns are measured using the same scale i.e., all columns have the same number of theoretical levels, even though some levels for some species may not be observed. The number of levels is then assumed to be given by the maximum value from all the ordinal columns of the response matrix. Because of this, all ordinal columns then assumed to have the *same* cutoffs, $\boldsymbol{\tau}$, while the response-specific intercept, β_{0j}, allows for deviations away from these common cutoffs. That is,

$$\Phi(P(y_{ij} \leq k)) = \tau_k + \beta_{0j} + \ldots,$$

where $\Phi(\cdot)$ is the probit function, $P(y_{ij} \leq k)$ is the cumulative probability of element y_{ij} being less than or equal to level k, τ_k is the cutoff linking levels k and $k + 1$ (and which are increasing in k), β_{0j} are the column effects, and $\ldots$ denotes what else is included in the model, e.g. latent variables and related coefficients. To ensure model identifiability, and also because they are interpreted as

response-specific deviations from the common cutoffs, the β_{0j}'s are treated as random effects and drawn from a normal distribution with mean zero and unknown standard deviation.

The parameterization above is useful for modeling ordinal in ecology. When ordinal responses are recorded, usually the same scale is applied to all species e.g., level 1 = not there, level 2 = a bit there, level 3 = lots there, level 4 = everywhere! The quantity τ_k can thus be interpreted as this common scale, while β_{0j} allows for deviations away from these to account for differences in species prevalence. Admittedly, the current implementation of boral for ordinal data can be quite slow.

For count distributions where zeros are not permitted, then the zero truncated Poisson (`family = "ztpoisson"`) and zero truncated negative binomial distributions (`family = "ztnegative.binomial"`) are possible. Note for these two distributions, and as is commonly implemented in other regression models e.g., the `countreg` package (Zeileis and Kleiber, 2018), the models are set up such that a log-link connects the mean of the *untruncated* distribution to the linear predictor. While not necessarily useful on its own, the zero truncated distributions may often be used in conjunction with an model for modeling presence-absence data, and together they can be used to construct the hurdle class of models (noting direct implementation of hurdle models is currently not available).

Finally, in the event different responses are collected for different columns, e.g., some columns of the response matrix are counts, while other columns are presence-absence, one can specify different distributions for each column. Aspects such as variable selection, residual analysis, and plotting of the latent variables are, in principle, not affected by having different distributions. Naturally though, one has to be more careful with interpretation of the row effects α_i and latent variables $\boldsymbol{u}_i$, as different link functions will be applied to each column of the response matrix. A situation where different distributions may prove useful is when the response matrix is a species–traits matrix, where each row is a species and each column a trait such as specific leaf area. In this case, traits could be of different response types, and the goal perhaps is to perform unconstrained ordination to look for patterns between species on an underlying trait surface e.g., a defense index for a species (Moles et al., 2013).

Warnings

- MCMC with lots of ordinal columns take an especially long time to run! Moreover, estimates for the cutoffs in cumulative probit regression may be poor for levels with little data. Major apologies for this advance =(

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Agresti, A. (2010). Analysis of Ordinal Categorical Data. Wiley.
- Foster, S. D. and Bravington, M. V. (2013). A Poisson-Gamma model for analysis of ecological non-negative continuous data. Journal of Environmental and Ecological Statistics, 20, 533-552.
- Korhonen, L., et al. (2007). Local models for forest canopy cover with beta regression. Silva Fennica, 41, 671-685.

- Moles et al. (2013). Correlations between physical and chemical defences in plants: Trade-offs, syndromes, or just many different ways to skin a herbivorous cat? New Phytologist, 198, 252-263.
- Smithson, M., and Verkuilen, J. (2006). A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. Psychological methods, 11, 54-71.
- Zeileis, A., and Kleiber C. (2018). countreg: Count Data Regression. R package version 0.2-1. URL http://R-Forge.R-project.org/projects/countreg/

See Also

`boral` for the main boral fitting function.

Examples

```
## Please see main boral function for examples.
```

`about.lvs` *Correlation structure for latent variables*

Description

[Stable]

This help file provides more information how (non-independence) correlation structures can be assumed for latent variables.

Details

In the main boral function, when latent varaibles are included, the default option is to assume that the latent variables are independent across the rows (sites) of the response matrix i.e., `lv.type = "independent"`. That is, $\boldsymbol{u}_i \sim N(\mathbf{0}, \boldsymbol{I}_d)$ where `d = num.lv`. This is useful when we want to model between species correlations (is a parsimonious manner), but it does make an assumption that sites are independent.

If one *a-priori* believes that the sites are, in fact, correlated e.g., due to spatial correlation, and that it cannot be sufficiently well accounted for by row effects (see comment below), then we can account for this by assuming a non-independence correlation structure for the the latent variables across sites. Note however we continue to assume that the d latent variables are still independent of one another. That is, if we let $\boldsymbol{u}_i = (u_{i1}, \ldots, u_{id})$, then we assume that for $l = 1, \ldots, d$,

$$(u_{1l}, u_{2l}, \ldots, u_{nl}) \sim N(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\Sigma}$ is some correlation matrix. When $\boldsymbol{\Sigma} = \boldsymbol{I}_n$ then we are back in the independence case. However, if we allow for the off-diagonals to be non-zero, then we the latent variables to be correlated, $\Sigma_{ij} = Cov(u_{il}, u_{jl})$. This in turn induces correlation across sites and species i.e., two species at two different sites are now correlated because of the correlation across sites.

While there are fancier structures and attempts at accounting for correlations between sites (Cressie and Wikle, 2015), in boral we assume relatively simple structures. Specifically, we can assume that

sites further away are less correlated, and so Σ can be characterized based on a distance matrix `distmat` and associated spatial covariance parameters which require estimation. Indeed, such simple spatial latent variable models have become rather popular in community ecology of late, at least as a first attempt at accounting for spatial (and also temporal) correlation e.g., Thorson et al., (2015, 2016); Ovaskainen et al., (2016, 2017).

At the moment, several correlation structures are permitted. Let D_{ij} denote the distance between site i and j i.e., entry (i, j) in `distmat`. Also, let $(\vartheta_1, \vartheta_2)$ denote the two spatial covariance parameters (noting that the second parameter is not required for some of structures). Then we have: 1) `lv.type = "exponential"` such that $\Sigma_{ij} = \exp(-D_{ij}/\vartheta_1)$; 2) `lv.type = "squared.exponential"`, such that $\Sigma_{ij} = \exp(-D_{ij}/\vartheta_1^2)$; 3) `lv.type = "power.exponential"`, such that $\Sigma_{ij} = \exp(-(D_{ij}/\vartheta_1)^{\vartheta_2})$ where $\vartheta_1 \in (0, 2]$; 4) `lv.type = "spherical"`, such that $(D_{ij} < \vartheta_1) * (1 - 1.5 * D_{ij}/\vartheta_1 + 0.5 * (D_{ij}/\vartheta_1)^3)$. We refer the reader to the `geoR` and the function `cov.spatial` for more, simple information on spatial covariance functions (Ribeiro Jr and Diggle, 2016).

It is important to keep in mind that moving away from an independence correlation structure for the latent variables *massively* increases computation time for MCMC sampling (and indeed any estimation method for latent variable models). Given JAGS is not the fastest of methods when it comes to MCMC sampling, then one should be cautious about moving away from indepndence. For example, if you *a-priori* have a nested experimental design which is inducing spatial correlation, then it is much faster and more effective to include (multiple) row effects in the model to account for this spatial correlation instead.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Cressie, N. and Wikle, C. K. (2015) Statistics for Spatio-temporal Data. John Wiley & Sons.
- Ovaskainen et al. (2016). Uncovering hidden spatial structure in species communities with spatially explicit joint species distribution models. Methods in Ecology and Evolution, 7, 428-436.
- Ovaskainen et al. (2017). How to make more out of community data? A conceptual framework and its implementation as models and software. Ecology Letters, 20, 561-576.
- Ribeiro Jr, P. J., and Diggle P. J., (2016). geoR: Analysis of Geostatistical Data. R package version 1.7-5.2. https://CRAN.R-project.org/package=geoR.
- Thorson et al. (2016). Joint dynamic species distribution models: a tool for community ordination and spatio-temporal monitoring. Global Ecology and Biogeography, 25, 1144-1158
- Thorson et al. (2015). Spatial factor analysis: a new tool for estimating joint species distributions and correlations in species range. Methods in Ecology and Evolution, 6, 627-63

See Also

`boral` for the main boral fitting function.

Examples

```
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The example below is taken directly from the boral help file

example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")

## Not run:
## Example 2d - model with environmental covariates and
##  two structured latent variables using fake distance matrix
fakedistmat <- as.matrix(dist(1:n))
spiderfit_lvstruc <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2, type = "exponential", distmat = fakedistmat),
     mcmc.control = example_mcmc_control, model.name = testpath)

summary(spiderfit_lvstruc)

## End(Not run)
```

---

about.ranefs *Including response-specific random intercepts in boral*

Description

[Stable]

This help file provides more information regarding the how response-specific random intercepts can be included to account for sampling design, induce correlation betweens observational units that are different to each response etc...

Details

As of version 2.0, it is now possible to include response-specific random intercepts in a model. There may be a number of reasons why a user may be to do this, but the most common reasons (at least in community ecology) would be to account sampling design on a response-specific basis, and more generally if there *a-priori* knowledge of clustering between observational units and so random intercepts are to be included to account for such potential within-cluster correlation.

Alternatively, if you are familiar with including random row effects in a model, then one may think of response-specific random intercepts as similar to this, except the row effects are now response-specific specific rather than a common across all responses. In doing so, response-specific random

intercepts are more flexible and allow the induced correlations (and thus the standard deviations) to be on a per-response basis, although this comes at the expense of some random intercepts being poorly estimates for responses with relatively little information in their observations e.g., rare species.

In for example, the formulation for the correlated response model

$$g(\mu_{ij}) = \alpha_i + \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j; \quad i = 1, \ldots, n; j = 1, \ldots, p,$$

the $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ denote response-specific random intercepts included, with bmb_j denoting the response-specific random intercepts, and $\boldsymbol{z}\top_i$ denoting the corresponding design vector for observational unit i, and are "constructed" based on the input `ranef.ids`.

Akin to other packages such as `lme4` or `glmmTMB`, all random intercepts are assumed to be normally distributed with the corresponding variance components are assumed to be response-specific. In fact, if we consider the simpler independent response model with no row effects

$$g(\mu_{ij}) = \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j; \quad i = 1, \ldots, n; j = 1, \ldots, p,$$

then the above would be equivalent to fitting a generalized linear mixed model (GLMM) to each response separately, for which packages such as `lme4` and `glmmTMB` are well suited for. In that sense, `boral` can fit some models, but can also incorporate row effects and/or latent variables to induce correlation between responses.

Perhaps not surprisingly, the way response-specific random intercepts are included is very similar to how row effects are included in the model. Specifically, the argument `ranef.ids` identifies the number of random intercepts to be included and how each observational unit maps to a cluster. `ranefs.ids` is a matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j. Examples of its use are provided in the help file below.

After the model is fitted, for each set of random intercepts included (i.e., each column of `ranef.ids`), estimates along with HPD intervals of the response-specific standard deviations of the corresponding random effects distributions, as well of the random intercept predictions are returned. These can then be visualized for example using the `ranefsplot` function, or analyzed as appropriately by the user.

Warnings

- It is usually not recommended to have both random row effects and response-specific random intercepts simultaneously in the same model, unless they are were at different levels of of the data)

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

boral for the main boral fitting function, ranefsplot for horizontal line or "caterpillar plot" of the response-specific random effects predictons (if applicable).

Examples

```
## Not run:
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The example below is taken directly from the boral help file

example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")

## Example 2e - Similar to 2c, but we will species-specific random intercepts
##   for the seven regions (with row effects in the model)
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$ranef.coefs.median
spiderfit_nb$ranef.sigma.median

## End(Not run)
```

---

about.ssvs *Stochastic search variable selection (SSVS) in boral*

Description

[Stable]

This help file provides more information regarding the implementation of the stochastic search variable selection (SSVS, George and McCulloch, 1993) as implemented in the boral package.

Details

Stochastic search variable selection (SSVS, George and McCulloch, 1993) is a approach for model selection, which is applicable specifically to the Bayesian MCMC framework. As of boral version 1.5, SSVS is implemented in two ways.

SSVS on coefficients in the covariate matrix $\boldsymbol{X}$: SSVS is implemented on the response-specific coefficients $\boldsymbol{\beta}_j$. Basically, SSVS works by placing a spike-and-slab priors on these coefficients,

such that the spike is a narrow normal distribution concentrated around zero and the spike is a normal distribution with a large variance.

$$\rho(\beta) = I_{\beta=1} \times \mathcal{N}(0, \sigma^2) + (1 - I_{\beta=1}) \times \mathcal{N}(0, g * \sigma^2),$$

where σ^2 is determined by `prior.control$hypparams[3]`, g is determined by `ssvs.g`, and $I_{\beta=1} = P(\beta = 1)$ is an indicator function representing whether coefficient is included in the model. It is given a Bernoulli prior with probability of inclusion 0.5. After fitting, the posterior probability of β being included in the model is returned based on posterior mean of the indicator function $I_{\beta=1}$. Note this is NOT the same as a *p*-value seen in maximum likelihood estimation: a *p*-value provides an indication of how much evidence there is against the null hypothesis of $\beta = 0$, while the posterior probability provides a measure of how likely it is for $\beta \neq 0$ given the data.

SSVS can be applied at a grouped or individual coefficient level, and this is governed by `prior.control$ssvs.index`:

- For elements of `ssvs.index` equal to -1, SSVS is not applied on the corresponding covariate of $\boldsymbol{X}$.
- For elements equal to 0, SSVS is applied to each individual coefficients of the corresponding covariate in $\boldsymbol{X}$. That is, the fitted model will return posterior probabilities for this covariate, one for each column of the response matrix.
- For elements taking positive integers 1, 2, and so on, SSVS is applied to each group of coefficients of the corresponding covariate in $\boldsymbol{X}$. That is, the fitted model will return a single posterior probability for this covariate, indicating whether this covariate should be included for all columns of the response matrix; see O'Hara and Sillanpaa (2009) and Tenan et al. (2014) among many others for an discussion of Bayesian variable selection methods.

Note the last application of SSVS allows multiple covariates to be selected *simultaneously*. For example, suppose the covariate matrix consists of five columns: the first two columns are environmental covariates, while the last three correspond to quadratic terms of the two covariates as well as their interaction. If we want to "test" whether any quadratic terms are required, then we can set `prior.control$ssvs.index = c(-1,-1,1,1,1)`, so a single posterior probability of inclusion is returned for the last three columns of the covariate matrix.

Finally, note that summaries such as posterior medians and HPD intervals of the coefficients, as well as performing residual analysis, from a fitted model that has implemented SSVS may be problematic because the posterior distribution is by definition multi-modal. It may be advisable instead to separate out their application of SSVS and posterior inference.

SSVS on trait coefficients: If traits are included in boral, thereby leading to a fourth corner model (see `about.traits` for more details on this type of model), SSVS can also be performed on the associated trait coefficients. That is, in such model we have

$$\beta_{0j} \sim N(\kappa_{01} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_1, \sigma_1^2)$$

for the response-specific intercepts, and

$$\beta_{jk} \sim N(\kappa_{0k} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_k, \sigma_k^2)$$

for $k = 1, \ldots, d$ where `d = ncol(X)`. Then if the a particular index in the argument `prior.control$ssvs.traitsindex` is set to 0, SSVS is performed on the corresponding element

in $\boldsymbol{\kappa}_1$ or $\boldsymbol{\kappa}_k$. For example, suppose `which.traits[[2]] = c(2,3)`, meaning that the β_{j1}'s are drawn from a normal distribution with mean depending only on the second and third columns of the trait matrix. Then
`prior.control$ssvs.traitsindex[[2]] = c(0,1)`, then a spike-and-slab prior is placed on the first coefficent in $\boldsymbol{\kappa}_2$, while the second coefficient is assigned the "standard" prior governed by the `prior.control$hypparams`. That is, SSVS is performed on the first but not the second coefficient in $\boldsymbol{\kappa}_2$.

Please keep in mind that because boral allows the user to manually decide which traits drive which covariates in $\boldsymbol{X}$, then care must be taken when setting up both `which.traits` and `prior.control$ssvs.traitsindex`. That is, when supplied then both objects should be lists of have the same length, and the length of the corresponding vectors comprising each element in the two lists should match as well e.g., `which.traits[[2]]` and `prior.control$ssvs.traitsindex[[2]]` should be of the same length.

Warnings

- Summaries of the coefficients such as posterior medians and HPD intervals may also be problematic when SSVS is being used, since the posterior distribution will be multi-modal.
- If `save.model = TRUE`, the raw jags model is also returned. This can be quite very memory-consuming, since it indirectly saves all the MCMC samples.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- George, E. I. and McCulloch, R. E. (1993). Variable selection via Gibbs sampling. Journal of the American Statistical Association, 85, 398-409.
- O'Hara, B., and Sillianpaa, M.J. (2009). A Review of Bayesian Variable Selection Methods: What, How and Which. Bayesian Analysis, 4, 85-118.
- Tenan et al. (2014). Bayesian model selection: The steepest mountain to climb. Ecological Modelling, 283, 62-69.

See Also

`boral` for the main boral fitting function which implementing SSVS, and `about.traits` for how fourth corner models work before applying SSVS to them.

Examples

```
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)
```

```
## NOTE: The two examples below and taken directly from the boral help file

example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Not run:
## Example 3a - Extend example 2 to demonstrate grouped covariate selection
## on the last three covariates.
example_prior_control <- list(type = c("normal","normal","normal","uniform"),
     ssvs.index = c(-1,-1,-1,1,2,3))
spiderfit_nb2 <- boral(y, X = X, family = "negative.binomial",
    mcmc.control = example_mcmc_control, prior.control = example_prior_control,
    model.name = testpath)

summary(spiderfit_nb2)


## Example 3b - Extend example 2 to demonstrate individual covariate selection
## on the last three covariates.
example_prior_control <- list(type = c("normal","normal","normal","uniform"),
     ssvs.index = c(-1,-1,-1,0,0,0))
spiderfit_nb3 <- boral(y, X = X, family = "negative.binomial",
    mcmc.control = example_mcmc_control, prior.control = example_prior_control,
    model.name = testpath)
summary(spiderfit_nb3)


## Example 5a - model fitted to count data, no site effects, and
## two latent variables, plus traits included to explain environmental responses
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)
## Just for fun, the regression coefficients for the second column of X,
## corresponding to the third element in the list example_which_traits,
## will be estimated separately and not regressed against traits.
example_which_traits[[3]] <- 0

fit_traits <- boral(y, X = X, traits = traits,
    which.traits = example_which_traits, family = "negative.binomial",
    mcmc.control = example_mcmc_control, model.name = testpath,
    save.model = TRUE)

summary(fit_traits)
```

```
## Example 5b - perform selection on trait coefficients
ssvs_traitsindex <- vector("list",ncol(X)+1)
for(i in 1:length(ssvs_traitsindex))
     ssvs_traitsindex[[i]] <- rep(0,ncol(traits))
ssvs_traitsindex[[3]] <- -1
fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", mcmc.control = example_mcmc_control,
    save.model = TRUE, prior.control = list(ssvs.traitsindex = ssvs_traitsindex),
    model.name = testpath)

summary(fit_traits)

## End(Not run)
```

`about.traits` *Including species traits in boral*

Description

[Stable]

This help file provides more information regarding the how species can be included to help mediate environmental responses, analogous to the so-called fourth corner problem.

Details

In the main boral function, when covariates $\boldsymbol{X}$ are included i.e. both the independent and correlated response models, one has the option of also including traits to help explain differences in species environmental responses to these covariates. Specifically, when a trait matrix is supplied, along with `which.traits`, then the β_{0j}'s and $\boldsymbol{\beta}_j$'s are then regarded as random effects drawn from a normal distribution. For the response-specific intercepts, we have

$$\beta_{0j} \sim N(\kappa_{01} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_1, \sigma_1^2),$$

where $(\kappa_{01}, \boldsymbol{\kappa}_1)$ are the regression coefficients relating to the traits to the intercepts and σ_1 is the error standard deviation. These are now the "parameters" in the model, in the sense that priors are assigned to them and MCMC sampling is used to estimate them (see the next section on estimation).

In an analogous manner, each of the elements in $\boldsymbol{\beta}_j = (\beta_{j1}, \ldots, \beta_{jd})$ are now drawn as random effects from a normal distribution. That is, for $k = 1, \ldots, d$ where `d = ncol(X)`, we have,

$$\beta_{jk} \sim N(\kappa_{0k} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_k, \sigma_k^2),$$

Which traits are to included (regressed) in the mean of the normal distributions is determined by the list argument `which.traits` in the main boral function. The first element in the list applies to $beta_{0j}$, while the remaining elements apply to the the $\boldsymbol{\beta}_j$. Each element of `which.traits` is a vector indicating which traits are to be used. For example, if `which.traits[[2]] = c(2,3)`, then the β_{j1}'s are drawn from a normal distribution with mean depending only on the second and third

columns of the trait matrix. If `which.traits[[2]][1] = 0`, then the regression coefficients are treated as independent, i.e. the values of β_{j1} are given their own priors and estimated separately from each other.

Including species traits in the model can be regarded as a method of simplifying the model: rather than each to estimates p sets of response-specific coefficients, we instead say that these coefficients are linearly related to the corresponding values of their traits (Warton et al., 2015; Ovaskainen et al., 2017). That is, we are using trait data to help explain similarities/differences in species responses to the environment. This idea has close relations to the fourth corner problem in ecology (Brown et al., 2014). Unlike the models of Brown et al. (2014) however, which treat the β_{0j}'s and β_{jk}'s are fixed effects and fully explained by the traits, boral adopts a random effects approach similar to Jamil et al. (2013) to "soak up" any additional between species differences in environmental responses not explained by traits.

Finally, note that from boral version 1.5, stochastic search variable selection (SSVS) can now be applied to the trait coefficients $\boldsymbol{\kappa}_1$ and $\boldsymbol{\kappa}_k$; please see `about.ssvs` for more details.

Warnings

- *No* intercept column should be included in the trait matrix, as it is included automatically.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Brown et al. (2014). The fourth-corner solution - using predictive models to understand how species traits interact with the environment. Methods in Ecology and Evolution, 5, 344-352.
- Jamil, et al. (2013). Selecting traits that explain species-environment relationships: a generalized linear mixed model approach. Journal of Vegetation Science 24, 988-1000
- Ovaskainen, et al. (2017). How to make more out of community data? A conceptual framework and its implementation as models and software. Ecology Letters, 20, 561-576.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, 30, 766-779.

See Also

`boral` for the main boral fitting function, and `about.ssvs` for implementing SSVS on fourth corner models.

Examples

```
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)
```

```
## NOTE: The two examples below and taken directly from the boral help file

example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")

## Not run:
## Example 5a - model fitted to count data, no site effects, and
## two latent variables, plus traits included to explain environmental responses
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)
## Just for fun, the regression coefficients for the second column of X,
## corresponding to the third element in the list example_which_traits,
## will be estimated separately and not regressed against traits.
example_which_traits[[3]] <- 0

fit_traits <- boral(y, X = X, traits = traits,
    which.traits = example_which_traits, family = "negative.binomial",
    mcmc.control = example_mcmc_control, model.name = testpath,
    save.model = TRUE)

summary(fit_traits)


## Example 5b - perform selection on trait coefficients
ssvs_traitsindex <- vector("list",ncol(X)+1)
for(i in 1:length(ssvs_traitsindex)) ssvs_traitsindex[[i]] <- rep(0,ncol(traits))
ssvs_traitsindex[[3]] <- -1
fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", mcmc.control = example_mcmc_control,
    save.model = TRUE, prior.control = list(ssvs.traitsindex = ssvs_traitsindex),
    model.name = testpath)

summary(fit_traits)


## Example 6 - simulate Bernoulli data, based on a model with two latent variables,
## no site variables, with two traits and one environmental covariates
## This example is a proof of concept that traits can used to
## explain environmental responses
library(mvtnorm)

n <- 100; s <- 50
X <- as.matrix(scale(1:n))
colnames(X) <- c("elevation")
```

```
traits <- cbind(rbinom(s,1,0.5), rnorm(s))
## one categorical and one continuous variable
colnames(traits) <- c("thorns-dummy","SLA")

simfit <- list(true.lv = rmvnorm(n, mean = rep(0,2)),
    lv.coefs = cbind(rnorm(s), rmvnorm(s, mean = rep(0,2))),
    traits.coefs = matrix(c(0.1,1,-0.5,1,0.5,0,-1,1), 2, byrow = TRUE))
rownames(simfit$traits.coefs) <- c("beta0","elevation")
colnames(simfit$traits.coefs) <- c("kappa0","thorns-dummy","SLA","sigma")

simy = create.life(true.lv = simfit$true.lv, lv.coefs = simfit$lv.coefs, X = X,
    traits = traits, traits.coefs = simfit$traits.coefs, family = "binomial")


example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)
fit_traits <- boral(y = simy, X = X, traits = traits,
    which.traits = example_which_traits, family = "binomial",
    lv.control = list(num.lv = 2), save.model = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)

## End(Not run)
```

`boral` *Fitting boral (Bayesian Ordination and Regression AnaLysis) models*

Description

[Stable]

Bayesian ordination and regression models for analyzing multivariate data in ecology. Three "types" of models may be fitted: 1) With covariates and no latent variables, boral fits independent response GLMs; 2) With no covariates, boral fits a pure latent variable model; 3) With covariates and latent variables, boral fits correlated response GLMs.

Usage

```
boral(y, ...)

## Default S3 method:
boral(y, X = NULL, formula.X = NULL, X.ind = NULL,
     traits = NULL, which.traits = NULL, family, trial.size = 1,
     lv.control = list(num.lv = 0, type = "independent", distmat = NULL),
     row.eff = "none", row.ids = NULL, ranef.ids = NULL,
     offset = NULL, save.model = FALSE, calc.ics = FALSE,
     mcmc.control = list(n.burnin = 10000, n.iteration = 40000,
     n.thin = 30, seed = NULL),
```

```
     prior.control = list(type = c("normal","normal","normal","uniform"),
     hypparams = c(10, 10, 10, 30), ssvs.index = -1, ssvs.g = 1e-6,
    ssvs.traitsindex = -1), do.fit = TRUE, model.name = NULL, num.lv = NULL, ...)

## S3 method for class 'boral'
print(x, ...)
```

Arguments

y	A response matrix of multivariate data e.g., counts, binomial or Bernoulli responses, continuous response, and so on. With multivariate abundance data ecology for instance, rows correspond to sites and columns correspond to species. Any categorical (multinomial) responses **must** be converted to integer values. For ordinal data, the minimum level of the response matrix must be 1 instead of 0.
X	Either a model matrix of covariates (otherwise known as the covariate matrix), which is included as part of the model, or a data frame from which the argument `formula.X` uses to create covariate matrix. Defaults to `NULL`, in which case no model matrix (and thus covariates) is included in the model. No intercept column should be included.
X.ind	An matrix of 1s and 0s, indicating whether a particular covariate should be included (1) or excluded (0) in the mean structure of a particular response. The matrix should the number of rows equal to the number of columns in the response matrix, and the number of columns equal to the number of columns in the covariate matrix Defaults to `NULL`, in which case it is assumed that all covariates are included in the mean structure of all responses i.e., all 1s.
formula.X	an object of class "formula", which represents a symbolic description of the covariate matrix to be created (based on the this argument along with the `X`, which will be treated as a data frame). The symbolic description works in the same way that standard regressions functions such as `lm` and `glm` work. The "-1" symbolic description, which denotes to not include an intercept, should not be included. Also, there should be nothing on the left hand side of the "~". Note that if this argument is supplied, then after fitting boral will return an `X` output which is not the original `X` supplied but the covariate matrix created as a consequence of this argument.
x	An object for class "boral".
traits	A model matrix of species traits (otherwise known as the trait matrix), which can be included as part of the model. Defaults to `NULL`, in which case no matrix was used. No intercept column should be included in the trait matrix, as it is included automatically.
which.traits	A list of length equal to (number of predictor variables in the model as implied by `X` and `formula.X`, plus 1), informing which columns of the trait matrix the response-specific intercepts and each of the response-specific regression coefficients should be regressed against. The first element in the list applies to the response-specific intercept, while the remaining elements apply to the regression coefficients. Each element of `which.traits` is a vector indicating which traits are to be used.

For example, if `which.traits[[2]] = c(2,3)`, then the regression coefficients corresponding to the first column in the covariate matrix are regressed against the second and third columns of the trait matrix. If `which.traits[[2]][1] = 0`, then the regression coefficients for each column are treated as independent. Please see `about.traits` for more details.

Defaults to `NULL`, and used in conjunction with `traits` and `prior.control$ssvs.traitsindex`.

`family` Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link).

Please see `about.distributions` for information on distributions available in boral overall.

`trial.size` Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

`lv.control` A list (currently) with the following arguments:

- *num.lv:* which specifies the number of true latent variables to generate. Defaults to 0.
- *type:* which specifies the type the correlation structure of the latent variables (across sites). Defaults to independence correlation structure.
- *distmat:* which a distance matrix required to calculate correlations across sites when a non-independence correlation structure on the latent variables is imposed.

Please see `about.lvs` for more information.

`row.eff` Single element indicating whether (multiple) row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and unknown variance, analogous to a random intercept in mixed models. Defaults to "none".

`row.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j. This matrix is useful if one wants to specify more complicated row effect structures beyond a single, row effect unique to each row; please see details below as well as examples below. Whether these

row effects are included as fixed or random effects is governed by `row.eff`. Defaults to `NULL`, so that if `row.eff = "none"` then the argument is ignored, otherwise if `row.eff = "fixed"` or `"random"`, then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row.

`ranef.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j; please see `about.ranefs` for details. Defaults to `NULL`, in which case it is assumed no random intercepts are to be included in the model. If supplied, then response-specific random intercepts are assumed to come from a normal distribution with mean zero and unknown (response-specific) standard deviation.

`offset` A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to `NULL`.

`save.model` If `save.model = TRUE`, then the JAGS model file is saved as a text file (with name given by `model.name`) in the current working directory as well as the MCMC samples, which themselves can be extracted using the `get.mcmcsamples` function. Various functions available in the `coda` package can be applied to the MCMC samples for diagnosing convergence. Note MCMC samples can take up a lot of memory. Defaults to `FALSE`.

`calc.ics` If `calc.ics = TRUE`, then various information criteria values are also returned, which could be used to perform model selection (see `get.measures`). Defaults to `FALSE`. WARNING: As of version 1.6, this function is longer updated!

`mcmc.control` A list of parameters for controlling the MCMC sampling. Values not set will assume default values. These include:

- *n.burnin:* Length of burnin i.e., the number of iterations to discard at the beginning of the MCMC sampler. Defaults to 10000.
- *n.iteration:* Number of iterations including burnin. Defaults to 40000.
- *n.thin:* Thinning rate. Must be a positive integer. Defaults to 30.
- *seed:* Seed for JAGS sampler. A `set.seed(seed)` command is run immediately before starting the MCMC sampler. Defaults to the value `NULL`, so the MCMC estimation is not seeded.

`prior.control` A list of parameters for controlling the prior distributions. Values not set will assume default values. These include:

- *type:* Vector of four strings indicating the type of prior distributions to use. In order, these are: 1) priors for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) priors for the latent variable coefficients. This is ignored if `num.lv = 0`; 3) priors for all response-specific coefficients relating to the covariate matrix (ignored if `X = NULL`). When traits are included in the model, this is also the prior for the trait regression coefficients (please see `about.traits` for more information); 4) priors for any dispersion parameters and variance (standard deviation, to be precise) parameters in the model.

For elements 1-3, the prior distributions currently available include: I) "normal", which is a normal prior with the variance controlled by elements 1-3 in hypparams; II) "cauchy", which is a Cauchy prior with variance controlled by elements 1-3 in hypparams. Gelman, et al. (2008) considers using Cauchy priors with variance 2.5^2 as weakly informative priors for coefficients in logistic and potentially other generalized linear models; III) "uniform", which is a symmetric uniform prior with minimum and maximum values controlled by element 1-3 in hypparams.
For element 4, the prior distributions currently available include: I) "uniform", which is uniform prior with minimum zero and maximum controlled by element 4 in hypparmas; II) "halfnormal", which is half-normal prior with variance controlled by hypparams; III) "halfcauchy", which is a half-Cauchy prior with variance controlled by element 4 in hypparams.
Defaults to the vector `c("normal","normal","normal","uniform")`.

- *hypparams:* Vector of four hyperparameters used in the set up of prior distributions. In order, these are: 1) affects the prior distribution for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) affects the prior distribution for all latent variable coefficients. This is ignored if `num.lv = 0`; 3) affects the prior distribution for response-specific coefficients relating to the covariate matrix (ignored if `X = NULL`). When traits are included in the model, it also affects the prior distribution for the trait regression coefficients; 4) affects the prior distribution for any dispersion parameters, as well as the prior distributions for the standard deviation of the random effects normal distribution if `row.eff = "random"`, the standard deviation of the response-specific random intercepts for these columns if more than two of the columns are ordinal, and the standard deviation of the random effects normal distribution for trait regression coefficients when traits are included in the model.
  Defaults to the vector `c(10, 10, 10, 30)`. The use of normal distributions with mean zero and variance 10 as priors is seen as one type of (very) weakly informative prior, according to Prior choice recommendations.
- *ssvs.index:* Indices to be used for stochastic search variable selection (SSVS, George and McCulloch, 1993). Either a single element or a vector with length equal to the number of columns in the covariate matrix. Each element can take values of -1 (no SSVS is performed on this covariate), 0 (SSVS is performed on individual coefficients for this covariate), or any integer greater than 0 (SSVS is performed on collectively all coefficients on this covariate.)
  Please see `about.ssvs` for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on the covariates.
- *ssvs.g:* Multiplicative, shrinkage factor for SSVS, which controls the strength of the "spike" in the SSVS mixture prior. In summary, if the coefficient is included in the model, the "slab" prior is a normal distribution with mean zero and variance given by element 3 in hypparams, while if the coefficient is not included in the model, the "spike" prior is normal distribution with mean zero and variance given by element 3 in hypparams multiplied by ssvs.g. Please see `about.ssvs` for more information regarding the imple-

mentation of SSVS. Defaults to 1e-6.

- *ssvs.traitsindex:* Used in conjunction with `traits` and `which.traits`, this is a list of indices to be used for performing SSVS on the trait coefficients. Should be a list with the same length as `which.traits`, and with each element a vector of indices with the same length as the corresponding element in `which.traits`. Each index either can take values of -1 (no SSVS on this trait coefficient) or 0 (no SSVS on this trait coefficient).
  Please see `about.ssvs` for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on any of the trait coefficients, if they are included in the model.

`do.fit` — If `do.fit = FALSE`, then only the JAGS model file is written to the current working directly (as text file with name based on `model.name`). No MCMC sampling is performed, and *nothing else* is returned. Defaults to `TRUE`.

`model.name` — Name of the text file that the JAGS script is written to. Defaults to `NULL`, in which case the default of "jagsboralmodel.txt" is used.

`num.lv` — Old argument superceded by `lv.control`. Defaults to `NULL` and ignored.

`...` — Not used.

Details

The boral package is designed to fit three types of models which may be useful in ecology (and probably outside of ecology as well =D).

Independent response models: boral allows explanatory variables to be entered into the model via the covariate matrix $\boldsymbol{X}$. If factors are to be included, then they should be parameterized using dummy variables. It should NOT include an intercept column. Alternatively, users can make use of the `formula.X` function to create the covariate model.

Without latent variables, i.e. `lv.control$num.lv = 0`, boral fits separate GLMs to each column of the $n \times p$ response matrix $\boldsymbol{Y}$, where the columns are assumed to be independent.

$$g(\mu_{ij}) = \alpha_i + \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j; \quad i = 1, \ldots, n; j = 1, \ldots, p,$$

where the mean response for element (i,j), denoted as mu_{ij}, is regressed against the covariates $\boldsymbol{x}_i$ via a link function $g(\cdot)$. The quantities $beta_{0j}$ and $\boldsymbol{beta}_j$ denote the response-specific intercepts and coefficients respectively, while `alpha_i` is an optional row effect that may be treated as a fixed or random effect, and $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ denote an optional set of response-specific random intercepts. Not all of these components may be included in the model, and the above is just representing the general case. If the included row effects are assumed to be fixed, then the first row effect is constrained to be zero for parameter identifiability reasons. If the include row effects are assumed to be random then they are drawn from a normal distribution with unknown variance ϕ^2. One reason we might want to include row effects is to account differences in sampling intensity between sites: these can lead to differences in site total abundance, and so by including fixed effects they play the same role as an offset to account for these differences.

boral can also handle multiple, hierarchical row effects, which may be useful to account for sampling design. This is controlled using the `row.ids` argument. For example, if the first five rows of y correspond to replications from site 1, the next five rows correspond to replications from site 2, and so on, then one can set `row.ids = matrix(c(1,1,1,1,1,2,2,2,2,2,...), ncol = 1)` to take

this in account. While this way of coding row effects via the row.ids argument takes some getting used to, it has been done this way partly to force the user to think more carefully about exactly the structure of the data i.e., with great power comes great responsibility...

Aside from row effects, and another more flexible way to account for sampling design but on a response-specific basis, boral allows users to include response-specific random intercepts. Please see about.ranefs for more information on this. Note response-specific random intercepts are permitted for all three types of models discussed. Akin to other packages such as lme4 or glmmTMB, all random intercepts are assumed to be normally distributed with the corresponding variance components are assumed to be response-specific.

If offset is supplied, then it will be included in the linear predictor below (and all linear predictors below where appropriate).

Without row effects, the above independent response model is basically a Bayesian analog of the manyglm function in the mvabund package (Wang et al., 2013). Note that X.ind argument can be optionally used to manually force certain covariates to be included in (1) or excluded from (0) from the mean structure of specific responses.

Pure latent variable models: If no explanatory variables are included and lv.control$num.lv > 0, boral fits a pure latent variable model to perform model-based unconstrained ordination (Hui et al., 2014),

$$g(\mu_{ij}) = \alpha_i + \beta_{0j} + \boldsymbol{z}_i^\top \boldsymbol{b}_j + \boldsymbol{u}\top_i \boldsymbol{\theta}_j,$$

where instead of measured covariates, we now have a vector of latent variables $\boldsymbol{u}_i$ with $\boldsymbol{\theta}_j$ being the response-specific coefficients relating to these latent variables. The response-specific intercept, β_{0j}, accounts for differences between species prevalence, while the row effect, $alpha_i$, is included to account for differences in site total abundance (typically assuming a fixed effect, row.eff = "fixed", although see Jamil and ter Braak, 2013, for a motivation for using random site effects), so that the ordination is then in terms of species composition. If α_i is omitted from the model i.e., row.eff = FALSE, then the ordination will be in terms of relative species abundance. As mentioned previously, one reason for including fixed row effects is to account of any potential differences in sampling intensity between sites.

As with the other types of models, α_i can be replaced with more sophisticated multiple, hierarchical row effects, and/or response-specific random intercepts given by $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ are optional.

Unconstrained ordination is used for visualizing multivariate data in a low-dimensional space, without reference to covariates (Chapter 9, Legendre and Legendre, 2012). Typically, lv.control$num.lv = 1 to 3 latent variables is used, allowing the latent variables to plotted (using lvsplot, for instance). The resulting plot can be interpreted in the same manner as plots from Nonmetric Multi-dimensional Scaling (NMDS, Kruskal, 1964) and Correspondence Analysis (CA, Hill, 1974), for example. A biplot can also be constructed by setting biplot = TRUE when using lvsplot, so that both the latent variables and their corresponding coefficients are plotted. For instance, with multivariate abundance data, biplots are used to visualize the relationships between sites in terms of species abundance or composition, as well as the indicator species for the sites.

Finally, boral offers a small number of options for allowing the latent variables to be correlated across rows of the responses. This may be useful when one has *a-priori* information about correlation between sites e.g., spatial correlation, which cannot be systematically accounted for by the inclusion of random effects (Thorson et al., 2015, 2016; Ovaskainen et al., 2016, 2017).. Please see the help file about.lvs for more information on this. By default, boral assumes the latent variables

are independent standard normally distributed across rows. Note the use of a non-independence correlation structure massively increases computation time!

Correlated response models: When one or more latent variables are included in conjunction with covariates i.e., a covariate matrix is supplied and `lv.control$num.lv` > 1, boral fits separate GLMs to each column of the response matrix $\boldsymbol{Y}$ while allowing for residual correlation between columns via the latent variables. This is quite useful for multivariate abundance data in ecology, where a separate GLM is fitted to species (modeling its response against environmental covariates), while accounting for the fact species at a site are likely to be correlated for reason other than similarities in environmental responses, e.g. biotic interaction, phylogeny, and so on. Correlated response model take the following form,

$$g(\mu_{ij}) = \alpha_i + \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j + \boldsymbol{u}_i^\top \boldsymbol{\theta}_j.$$

This model is thus a combinaton of the first two types of models. The linear predictor $\boldsymbol{u}_i^\top \boldsymbol{\theta}_j$ induces a residual covariance between the columns of the response matrix $\boldsymbol{y}$ (which is of rank `lv.control$num.lv`). For multivariate abundance data, this leads to a parsimonious method of accounting for correlation between species not due to the shared environmental responses. After fitting the model, the residual correlation matrix then can be obtained via the `get.residual.cor` function. Note `lv.control$num.lv` > 1 is necessarily in order to flexibly model the residual correlations; see Pollock et al. (2014) for residual correlation matrices in the context of Joint Species Distribution Models, Ovaskainen et al., (2017), and Warton et al. (2015, 2016) for an overview of latent variable models in multivariate ecology.

As with the other types of models, α_i can be replaced with more sophisticated multiple, hierarchical row effects, and/or response-specific random intercepts given by $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ are optional.

Distributions: A variety of families are available in boral, designed to handle multivariate abundance data of varying response types. Please see `about.distributions` for more information on this.

Including species traits: When covariates are included i.e. both the independent and correlated response models, one has the option of also including traits to help explain differences in species environmental responses to these covariates. Please see `about.traits` for more information on this.

Including response-specific random intercepts: For some types of data e.g. nested designs, it may be appropriate to include response-specific random intercepts. This can be considered as like row effects, but now the effects are specific to each response rather than the same across all responses. Please see `about.ranefs` for more information on this. Note as a consequence, it is usually not recommended to have both random row effects and response-specific random intercepts simultaneously in the same model (unless they are were at different levels of of the data).

Estimation: Estimation for all models is performed using Bayesian Markov Chain Monte Carlo (MCMC) methods via JAGS (Plummer, 2003). Please note that only *one* MCMC chain in run: this point is discussed later in this help file. Regarding prior distributions, the default settings, based on the `prior.control` argument, are as follows:

- Normal priors with mean zero and variance given by element 1 in `hypparams` are assigned to all intercepts, cutoffs for ordinal responses, and row effects.
- Normal priors with mean zero and variance given by element 2 in `hypparams` are assigned coefficients relating to latent variables, $\boldsymbol{\theta}_j$.

- Normal priors with mean zero and variance given by element 3 `hypparams` are assigned to all coefficients relating to covariates in $\boldsymbol{\beta}_j$. If traits are included, the same normal priors are assigned to the κ's, and the standard deviation σ_k are assigned uniform priors with maximum equal to element 4 in `hypparams`.
- For the negative binomial, normal, lognormal, and tweedie distributions, uniform priors with maximum equal to element 4 in `hypparams` are used on the dispersion parameters. Please note that for the normal and lognormal distributions, these uniform priors are assigned to the standard deviations ϕ (see Gelman, 2006). If there are any variance (standard deviation, to be precise) parameters in the model, then these are also assigned uniform priors with maximum equal to element 4 in `hypparams` e.g., standard deviation of the normal random effects if the row effects are assumed to random, the standard deviation of the normal random response-specific intercepts if more than two columns are ordinal responses, and the standard deviation of the normal random response-specific random intercepts when `ranef.ids` is supplied etc...

Using information criteria at your own risk: Using information criterion from `calc.ics = TRUE` for model selection should be done with extreme caution, for two reasons: 1) The implementation of some of these criteria is heuristic and experimental, 2) Deciding what model to fit should also be driven by the science and questions of interest. For example, it may be the case that a criterion suggests a model with 3 or 4 latent variables is more appropriate. However, if we are interested in visualizing the data for ordination purposes, then models with 1 or 2 latent variables are more appropriate. As another example, whether or not we include row effects when ordinating multivariate abundance data depends on if we are interested in differences between sites in terms of relative species abundance (`row.eff = "none"`) or species composition (`row.eff = "fixed"`). From version 1.6, the calculation of all information criteria is being gradually phased out!

SSVS: Stochastic search variable selection (SSVS, George and McCulloch, 1993) is also implemented for the response-specific coefficients $\boldsymbol{\beta}_j$. Please see `about.ssvs` for more information on this approach.

Value

An object of class "boral" is returned, being a list containing (but not limited to) the following components where applicable:

`call` The matched call.

`lv.coefs.mean/median/sd/iqr`
Matrices containing the mean/median/standard deviation/interquartile range of the posterior distributions of the latent variable coefficients. This also includes the response-specific intercepts, and dispersion parameters if appropriate.

`lv.mean/median/sd/iqr`
A matrix containing the mean/median/standard deviation/interquartile range of the posterior distributions of the latent variables.

`lv.covparams.mean/median/sd/iqr`
A matrix containing the mean/median/standard deviation/interquartile range of the posterior distributions for the parameters characterizing the correlation structure of the latent variables when they are assumed to be non-independent across rows.

`X.coefs.mean/median/sd/iqr`
: Matrices containing the mean/median/standard deviation/interquartile range of the posterior distributions of the response-specific coefficients relating to the covariate matrix.

`traits.coefs.mean/median/sd/iqr`
: Matrices containing the mean/median/standard deviation/interquartile range of the posterior distributions of the coefficients and standard deviation relating to the species traits; please see `about.traits`.

`cutoffs.mean/median/sd/iqr`
: Vectors containing the mean/median/standard deviation/interquartile range of the posterior distributions of the common cutoffs for ordinal responses (please see the not-so-brief tangent on distributions above).

`ordinal.sigma.mean/median/sd/iqr`
: Scalars containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the random intercept normal distribution corresponding to the ordinal response columns.

`powerparam.mean/median/sd/iqr`
: Scalars for the mean/median/standard deviation/interquartile range of the posterior distributions of the common power parameter for tweedie responses (please see the not-so-brief tangent on distributions above).

`row.coefs.mean/median/sd/iq`
: A list with each element containing the vectors of the mean/median/standard deviation/interquartile range of the posterior distributions of the row effects. The length of the list is equal to the number of row effects included i.e., `ncol(row.ids)`.

`row.sigma.mean/median/sd/iqr`
: A list with each element containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the row random effects normal distribution. The length of the list is equal to the number of row effects included i.e., `ncol(row.ids)`.

`ranef.coefs.mean/median/sd/iq`
: A list with each element containing the matrices of the mean/median/standard deviation/interquartile range of the posterior distributions of the response-specific random intercepts. The length of the list is equal to the number of random intercepts included i.e., `ncol(ranef.ids)`.

`ranef.sigma.mean/median/sd/iqr`
: A matrix containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the response-specific random intercept normal distribution. The number of columns of the matrix is equal to number of random intercepts included i.e., `ncol(ranef.ids)`.

`ssvs.indcoefs.mean/ssvs.indcoefs.sd`
: Matrices containing posterior probabilities and associated standard deviation for individual SSVS of coefficients in the covariate matrix.

`ssvs.gpcoefs.mean/ssvs.gpcoefs.sd`
: Matrices containing posterior probabilities and associated standard deviation for group SSVS of coefficients in the covariate matrix.

`ssvs.traitscoefs.mean/ssvs.traitscoefs.sd`
: Matrices containing posterior probabilities and associated standard deviation for individual SSVS of coefficients relating to species traits.

`hpdintervals` A list containing components which correspond to the lower and upper bounds of highest posterior density (HPD) intervals for all the parameters indicated above. Please see `get.hpdintervals` for more details.

`ics` If `calc.ics = TRUE`, then a list of different information criteria values for the model calculated using `get.measures` is run. Please see `get.measures` for details regarding the criteria. Also, please note the ics returned are based on `get.measures` with `more.measures = FALSE`.

`jags.model` If `save.model = TRUE`, the raw jags model fitted is returned. This can be quite large!

`geweke.diag` A list with two elements. The first element is itself a list containing the Geweke convergence diagnostic (Z-scores) for all appropriate parameters in the model. The second element contains the proportion of Z-scores that whose corresponding p-value is less than 0.05. No adjustment is made for multiple comparison on the p-values. Please see the section *Why is only one MCMC chain run?* for more information on this diagnostic.

`n`, `p`, `family`, `trial.size`, `num.lv`, `...` Various attributes of the model fitted, including the dimension of the response matrix, the response and model matrix used, distributional assumptions and trial sizes, number of latent variables, the number of covariates and traits, hyperparameters used in the Bayesian estimation, indices for SSVS, the number of levels for ordinal responses, and n.burnin, n.iteration and n.thin.

Why is only one MCMC chain run?

Much like the `MCMCfactanal` function in the `MCMCpack` package (Martin et al., 2011) for conducting factor analysis, which is a special case of the pure latent variable model with Gaussian responses, boral deliberately runs only one MCMC chain. This runs contrary to the recommendation of most Bayesian analyses, where the advice is to run multiple MCMC chains and check convergence using (most commonly) the Gelman-Rubin statistic (Gelman et al., 2013). The main reason for this is that, in the context of MCMC sampling, the latent variable model is invariant to a switch of the sign, i.e. $\boldsymbol{u}_i^\top \boldsymbol{\theta}_j = (-\boldsymbol{u}_i^\top (-\boldsymbol{\theta}_j)$, and so is actually unidentifiable.

As a result of sign-switching, different MCMC chains can produce latent variables and corresponding coefficients values that, while having similar magnitudes, will be different in sign. Consequently, combining MCMC chains and checking Rhats, computing posterior means and medians etc...becomes complicated (in principle, one way to resolve this problem would be to post-process the MCMC chains and deal with sign switching, but this is really hard!). Therefore, to alleviate this issue together, boral chooses to only run one MCMC chain.

What does this mean for the user?

- boral automatically calculates the Geweke convergence diagnostic (Geweke, 1992), which is a diagnostic applicable with only one MCMC chain; please see the help file `geweke.diag` in the `coda` package for more information. The output is a list containing Z-scores for the appropriate parameters in the model, and each score can be interpreted in the same manner as the test statistic from conducting a Z-test i.e., if the score exeeds roughly 1.96 then the p-value is less than 0.05, and there is evidence the MCMC chain (for this particular parameter) has not converged.

  The output from boral also provides the proportion of Z-scores whose corresponding p-values are less than 0.05. Of course, because there are a large number of parameters in the model,

then there are large number of Z-scores, and boral does not make any multiple comparison adjustment for this when calculating the number of "significant" Z-scores. If you do indeed want to use this diagnostic to formally check for convergence, then we recommend you conduct some adjustment e.g., using Holm's method, by doing something such as `gew.pvals <- 2*pnorm(abs(unlist(fit$geweke.diag[[1]])), lower.tail = FALSE)` and then `p.adjust(gew.pvals, method = "holm")`.

- For checking convergence, we recommend you look at trace plots of the MCMC chains. Using the `coda` package, which is automatically loaded when the `boral` package is loaded, try something like `plot(get.mcmcsamples(fit))`.
- If you have a lot of data, e.g. lots of sites compared to species, sign-switching tends to be less of problem and pops up less often.

Warnings

- *No* intercept column should be included in the covariate matrix. Column-specific intercepts are estimated automatically and given by the first column of `lv.coefs`. Similarly, *no* intercept column should be included in the trait matrix, as it is included automatically.
- As of version 1.6, functions to calculate information criteria along with `calc.marglogLik` are no longer updated, and being phrased out!
- MCMC with a non-independence correlation structure for the latent variables takes an especially long time to run! Likewise, MCMC with lots of ordinal columns take an especially long time to run! Moreover, estimates for the cutoffs in cumulative probit regression may be poor for levels with little data. Major apologies for this advance =(
- Summaries of the coefficients such as posterior medians and HPD intervals may also be problematic when SSVS is being used, since the posterior distribution will be multi-modal.
- If `save.model = TRUE`, the raw jags model is also returned. This can be quite very memory-consuming, since it indirectly saves all the MCMC samples.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Gelman A. (2006) Prior distributions for variance parameters in hierarchical models. Bayesian Analysis 1, 515-533.
- Gelman et al. (2008). A weakly informative default prior distribution for logistic and other regression models. The Annals of Applied Statistics, 2, 1360-1383.
- Gelman et al. (2013) Bayesian Data Analysis. CRC Press.
- George, E. I. and McCulloch, R. E. (1993). Variable selection via Gibbs sampling. Journal of the American Statistical Association, 85, 398-409.
- Geweke, J. (1992) Evaluating the accuracy of sampling-based approaches to calculating posterior moments. In Bayesian Statistics 4 (editors JM Bernado, JO Berger, AP Dawid and AFM Smith). Clarendon Press.

- Hui et al. (2014). Model-based approaches to unconstrained ordination. Methods in Ecology and Evolution, 6, 399-411.
- Hill, M. O. (1974). Correspondence analysis: a neglected multivariate method. Applied statistics, 23, 340-354.
- Jamil, T., and ter Braak, C.J.F. (2013). Generalized linear mixed models can detect unimodal species-environment relationships. PeerJ 1: e95.
- Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. Psychometrika, 29, 115-129.
- Legendre, P. and Legendre, L. (2012). Numerical ecology, Volume 20. Elsevier.
- Martin et al. (2011). MCMCpack: Markov Chain Monte Carlo in R. Journal of Statistical Software, 42, 1-21. URL: http://www.jstatsoft.org/v42/i09/.
- McLachlan, G., and Peel, D. (2004). Finite Mixture Models. Wiley.
- Ovaskainen, et al. (2016). Uncovering hidden spatial structure in species communities with spatially explicit joint species distribution models. Methods in Ecology and Evolution, 7, 428-436.
- Ovaskainen, et al. (2017). How to make more out of community data? A conceptual framework and its implementation as models and software. Ecology Letters, 20, 561-576.
- Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In Proceedings of the 3rd International Workshop on Distributed Statistical Computing. March (pp. 20-22).
- Pollock, et al. (2014). Understanding co-occurrence by modelling species simultaneously with a Joint Species Distribution Model (JSDM). Methods in Ecology and Evolution, 5, 397-406.
- Skrondal, A., and Rabe-Hesketh, S. (2004). Generalized latent variable modeling: Multilevel, longitudinal, and structural equation models. CRC Press.
- Thorson, et al. (2016). Joint dynamic species distribution models: a tool for community ordination and spatio-temporal monitoring. Global Ecology and Biogeography, 25, 1144-1158
- Thorson, et al. (2015). Spatial factor analysis: a new tool for estimating joint species distributions and correlations in species range. Methods in Ecology and Evolution, 6, 627-63
- Warton et al. (2012). Distance-based multivariate analyses confound location and dispersion effects. Methods in Ecology and Evolution, 3, 89-101.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, 30, 766-779.
- Warton et al. (2016). Extending joint models in community ecology: A response to Beissinger et al. Trends in ecology & evolution, 31, 737-738.
- Wang et al. (2013). `mvabund`: statistical methods for analysing multivariate abundance data. R package version 3.8.4.

See Also

`calc.varpart` to calculate variance partitioning of the covariates, `coefsplot` for horizontal line or "caterpillar plot" of the regression coefficients corresponding to the covariate matrix (if applicable),

get.enviro.cor and get.residual.cor for calculating the correlation matrix between the explanatory variables in the covariate matrix and the residual correlation matrix respectively, lvsplot for a scatter plot of the latent variables (and their coefficients if applicable), predict.boral for calculating predictions from a fitted model. ranefsplot for horizontal line or "caterpillar plot" of the response-specific random effects predictons (if applicable), summary.boral for a summary of the fitted model,

Examples

```
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1 - model with two latent variables, site effects,
##  and no environmental covariates
spiderfit_nb <- boral(y, family = "negative.binomial",
    lv.control = list(num.lv = 2), row.eff = "fixed",
    mcmc.control = example_mcmc_control, model.name = testpath)

summary(spiderfit_nb)

par(mfrow = c(2,2))
plot(spiderfit_nb) ## Plots used in residual analysis,
## Used to check if assumptions such an mean-variance relationship
## are adequately satisfied.

lvsplot(spiderfit_nb) ## Biplot of the latent variables,
## which can be interpreted in the same manner as an ordination plot.


## Not run:
## Example 2a - model with no latent variables, no site effects,
##      and environmental covariates
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
     mcmc.control = example_mcmc_control, model.name = testpath)
## Alternatively, you can use the formula.X argument for more custom
##   creation of a covariate matrix. For example, for have a covariate
##   including all predictors except bare sand:
# spiderfit_nb <- boral(y, X = X, formula.X = ~ . - bare.sand,
#     family = "negative.binomial", mcmc.control = example_mcmc_control,
#     model.name = testpath)
```

```
summary(spiderfit_nb)
## The results can be compared with the default example from
## the manyglm() function in mvabund.

## Caterpillar plots for the coefficients
par(mfrow=c(2,3), mar = c(5,6,1,1))
sapply(colnames(spiderfit_nb$X), coefsplot, object = spiderfit_nb)


## Example 2b - suppose now, for some reason, the 28 rows were
##  sampled such into four replications of seven sites
## Let us account for this as a fixed effect
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    row.eff = "fixed", row.ids = matrix(rep(1:7,each=4),ncol=1),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$row.coefs


## Example 2c - suppose now, for some reason, the 28 rows reflected
##  a nested design with seven regions, each with four sub-regions
## We can account for this nesting as a random effect
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    row.eff = "random",
    row.ids = cbind(1:n, rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$row.coef.median


## Example 2d - model with environmental covariates and
##  two structured latent variables using fake distance matrix
fakedistmat <- as.matrix(dist(1:n))
spiderfit_lvstruc <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2, type = "exponential", distmat = fakedistmat),
     mcmc.control = example_mcmc_control, model.name = testpath)

summary(spiderfit_lvstruc)


## Example 2e - Similar to 2c, but we will species-specific random intercepts
##   for the seven regions (with row effects in the model)
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$ranef.coefs.median
spiderfit_nb$ranef.sigma.median


## Example 3a - Extend example 2 to demonstrate grouped covariate selection
## on the last three covariates.
```

```
example_prior_control <- list(type = c("normal","normal","normal","uniform"),
     ssvs.index = c(-1,-1,-1,1,2,3))
spiderfit_nb2 <- boral(y, X = X, family = "negative.binomial",
    mcmc.control = example_mcmc_control,
    prior.control = example_prior_control, model.name = testpath)

summary(spiderfit_nb2)


## Example 3b - Extend example 2 to demonstrate individual covariate selection
## on the last three covariates.
example_prior_control <- list(type = c("normal","normal","normal","uniform"),
    ssvs.index = c(-1,-1,-1,0,0,0))
spiderfit_nb3 <- boral(y, X = X, family = "negative.binomial",
    mcmc.control = example_mcmc_control, prior.control = example_prior_control,
    model.name = testpath)
summary(spiderfit_nb3)


## Example 4 - model fitted to presence-absence data, no site effects, and
## two latent variables
data(tikus)
y <- tikus$abun
y[y > 0] <- 1
y <- y[1:20,] ## Consider only years 1981 and 1983
y <- y[,apply(y > 0,2,sum) > 2] ## Consider only spp with more than 2 presences

tikusfit <- boral(y, family = "binomial",
    lv.control = list(num.lv = 2), mcmc.control = example_mcmc_control,
    model.name = testpath)

lvsplot(tikusfit, biplot = FALSE)
## A strong location between the two sampling years


## Example 5a - model fitted to count data, no site effects, and
## two latent variables, plus traits included to explain environmental responses
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)
## Just for fun, the regression coefficients for the second column of X,
## corresponding to the third element in the list example_which_traits,
## will be estimated separately and not regressed against traits.
example_which_traits[[3]] <- 0

fit_traits <- boral(y, X = X, traits = traits,
    lv.control = list(num.lv = 2),
    which.traits = example_which_traits, family = "negative.binomial",
```

```
    mcmc.control = example_mcmc_control, model.name = testpath,
    save.model = TRUE)

summary(fit_traits)


## Example 5b - perform selection on trait coefficients
ssvs_traitsindex <- vector("list",ncol(X)+1)
for(i in 1:length(ssvs_traitsindex))
     ssvs_traitsindex[[i]] <- rep(0,ncol(traits))
ssvs_traitsindex[[3]] <- -1
fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial",
    lv.control = list(num.lv = 2), mcmc.control = example_mcmc_control,
    save.model = TRUE, prior.control = list(ssvs.traitsindex = ssvs_traitsindex),
    model.name = testpath)

summary(fit_traits)


## Example 6 - simulate Bernoulli data, based on a model with two latent variables,
## no site variables, with two traits and one environmental covariates
## This example is a proof of concept that traits can used to
## explain environmental responses
library(mvtnorm)

n <- 100; s <- 50
X <- as.matrix(scale(1:n))
colnames(X) <- c("elevation")

traits <- cbind(rbinom(s,1,0.5), rnorm(s))
## one categorical and one continuous variable
colnames(traits) <- c("thorns-dummy","SLA")

simfit <- list(true.lv = rmvnorm(n, mean = rep(0,2)),
    lv.coefs = cbind(rnorm(s), rmvnorm(s, mean = rep(0,2))),
    traits.coefs = matrix(c(0.1,1,-0.5,1,0.5,0,-1,1), 2, byrow = TRUE))
rownames(simfit$traits.coefs) <- c("beta0","elevation")
colnames(simfit$traits.coefs) <- c("kappa0","thorns-dummy","SLA","sigma")

simy = create.life(true.lv = simfit$true.lv, lv.coefs = simfit$lv.coefs, X = X,
    traits = traits, traits.coefs = simfit$traits.coefs, family = "binomial")


example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)
fit_traits <- boral(y = simy, X = X, traits = traits,
    which.traits = example_which_traits, family = "binomial",
    lv.control = list(num.lv = 2), save.model = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)
```

```
## End(Not run)
```

`calc.condlogLik` *Conditional log-likelihood for a fitted model*

Description

[Deprecated]

Calculates the conditional log-likelihood for a set of parameter estimates from a fitted model, where everything is treated as "fixed effects" including latent variables, row effects, and so on. WARNING: As of version 1.9, this function is no longer being maintained (and probably does not work properly, if at all)!

Usage

```
calc.condlogLik(y, X = NULL, family, trial.size = 1, lv.coefs,
 X.coefs = NULL, row.coefs = NULL, row.ids = NULL,
 offset = NULL, lv = NULL, cutoffs = NULL, powerparam = NULL)
```

Arguments

`y`	The response matrix the model was fitted to.
`X`	The covariate matrix used in the model. Defaults to `NULL`, in which case it is assumed no model matrix was used.
`family`	Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression). Please see `about.distributions` for information on distributions available in boral overall.
`trial.size`	Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.
`lv.coefs`	The response-specific intercept, coefficient estimates relating to the latent variables, and dispersion parameters from the fitted model.
`X.coefs`	The coefficients estimates relating to the covariate matrix from the fitted model. Defaults to `NULL`, in which it is assumed there are no covariates in the model.

`row.coefs` Row effect estimates for the fitted model. The conditional likelihood is defined conditional on these estimates i.e., they are also treated as "fixed effects". Defaults to `NULL`, in which case it is assumed there are no row effects in the model.

`row.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see the `boral` function for details. Defaults to `NULL`, so that if `row.coefs = NULL` then the argument is ignored, otherwise if `row.coefs` is supplied then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row. An internal check is done to see `row.coefs` and `row.ids` are consistent in terms of arguments supplied.

`offset` A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to `NULL`.

`lv` Latent variables "estimates" from the fitted model, which the conditional likelihood is based on. Defaults to `NULL`, in which case it is assumed no latent variables were included in the model.

`cutoffs` Common cutoff estimates from the fitted model when any of the columns of the response matrix are ordinal responses. Defaults to `NULL`.

`powerparam` Common power parameter from the fitted model when any of the columns of the response matrix are tweedie responses. Defaults to `NULL`.

Details

For an nxp response matrix $\boldsymbol{Y}$, suppose we fit a model with one or more latent variables. If we denote the latent variables by $\boldsymbol{u}_i; i = 1, \ldots, n$, then the conditional log-likelihood is given by,

$$\log(f) = \sum_{i=1}^{n} \sum_{j=1}^{p} \log\{f(y_{ij}|\boldsymbol{u}_i, \boldsymbol{\theta}_j, \beta_{0j}, \ldots)\},$$

where $f(y_{ij}|\cdot)$ is the assumed distribution for column j, $\boldsymbol{u}_i$ are the latent variables and $\boldsymbol{\theta}_j$ are the coefficients relating to them, β_{0j} are response-specific intercepts, and $\ldots$ denotes anything else included in the model, such as row effects, regression coefficients related the covariate matrix and the trait matrix, etc...

The key difference between this and the marginal likelihood (see `calc.marglogLik`) is that the conditional likelihood treats everything as "fixed effects" i.e., conditions on them. These include the latent variables $\boldsymbol{u}_i$ and other parameters that were included in the model as random effects e.g., row effects if `row.eff = "random"`, regression coefficients related to the covariate matrix if traits were included in the model, and so on.

The conditional DIC, WAIC, EAIC, and EBIC returned from `get.measures` are based on the conditional likelihood calculated from this function. Additionally, `get.measures` returns the conditional likelihood evaluated at all MCMC samples of a fitted model.

Value

A list with the following components:

logLik	Value of the conditional log-likelihood.
logLik.comp	A matrix of the log-likelihood values for each element in the response matrix, such that sum(logLik.comp) = logLik.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

calc.logLik.lv0 to calculate the conditional/marginal log-likelihood for a model with no latent variables; calc.marglogLik for calculation of the marginal log-likelihood;

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

## Example 1 - model with 2 latent variables, site effects,
##  and no environmental covariates
spiderfit_nb <- boral(y, family = "negative.binomial",
    lv.control = list(num.lv = 2), row.eff = "fixed",
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
site_coef <- list(ID1 = apply(fit_mcmc[,grep("row.coefs.ID1", mcmc_names)],
    2,median))
lvs_mat <- matrix(apply(fit_mcmc[,grep("lvs",mcmc_names)],2,median),nrow=n)

## Calculate the conditional log-likelihood at the posterior median
calc.condlogLik(y, family = "negative.binomial",
    lv.coefs = coef_mat, row.coefs = site_coef, lv = lvs_mat)
```

```
## Example 2 - model with no latent variables and environmental covariates
X <- scale(spider$x)
spiderfit_nb2 <- boral(y, X = X, family = "negative.binomial",
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb2)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
X_coef_mat <- matrix(apply(fit_mcmc[,grep("X.coefs",mcmc_names)],
    2,median),nrow=p)

## Calculate the log-likelihood at the posterior median
calc.condlogLik(y, X = X, family = "negative.binomial",
    lv.coefs =  coef_mat, X.coefs = X_coef_mat)

## End(Not run)
```

`calc.logLik.lv0` *Log-likelihood for a model fitted with no latent variables*

Description

[Deprecated]

Calculates the log-likelihood for a set of parameter estimates from a model with no latent variables. If the row effects are assumed to be random, they are integrated over using Monte Carlo integration. WARNING: As of version 1.9, this function is no longer being maintained (and probably does not work properly, if at all)!

Usage

```
calc.logLik.lv0(y, X = NULL, family, trial.size = 1, lv.coefs,
 X.coefs = NULL, row.eff = "none", row.params = NULL,
 row.ids = NULL, offset = NULL, cutoffs = NULL,
 powerparam = NULL)
```

Arguments

`y`	The response matrix the model was fitted to.
`X`	The covariate matrix used in the model. Defaults to `NULL`, in which case it is assumed no model matrix was used.

`family` Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression).

Please see `about.distributions` for information on distributions available in boral overall.

`trial.size` Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

`lv.coefs` The response-specific intercept, coefficient estimates relating to the latent variables, and dispersion parameters from the fitted model.

`X.coefs` The coefficients estimates relating to the covariate matrix from the fitted model. Defaults to `NULL`, in which it is assumed there are no covariates in the model.

`row.eff` Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and standard deviation given by `row.params`. Defaults to "none".

`row.params` Parameters corresponding to the row effect from the fitted model. If `row.eff = "fixed"`, then these are the fixed effects and should have length equal to the number of columns in the response matrix. If `row.eff = "random"`, then this is the standard deviation for the random effects normal distribution. If `row.eff = "none"`, then this argument is ignored.

`row.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see `boral` for details. Defaults to `NULL`, so that if `row.params = NULL` then the argument is ignored, otherwise if `row.params` is supplied then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row. An internal check is done to see `row.params` and `row.ids` are consistent in terms of arguments supplied.

`offset` A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to `NULL`.

`cutoffs` Common cutoff estimates from the fitted model when any of the columns of the response matrix are ordinal responses. Defaults to `NULL`.

`powerparam` Common power parameter from the fitted model when any of the columns of the response matrix are tweedie responses. Defaults to `NULL`.

Details

For an nxp response matrix $\boldsymbol{y}$, the log-likelihood for a model with no latent variables included is given by,

$$\log(f) = \sum_{i=1}^{n}\sum_{j=1}^{p} \log\{f(y_{ij}|\beta_{0j}, \alpha_i, \ldots)\},$$

where $f(y_{ij}|\cdot)$ is the assumed distribution for column j, β_{0j} is the response-specific intercepts, α_i is the row effect, and ... generically denotes anything else included in the model, e.g. row effects, dispersion parameters etc...

Please note the function is written conditional on all regression coefficients. Therefore, if traits are included in the model, in which case the regression coefficients $\beta_{0j}, \boldsymbol{\beta}_j$ become random effects instead (please see `about.traits`), then the calculation of the log-likelihood does NOT take this into account, i.e. does not marginalize over them!

Likewise if more than two columns are ordinal responses, then the regression coefficients β_{0j} corresponding to these columns become random effects, and the calculation of the log-likelihood also does NOT take this into account, i.e. does not marginalize over them!

When a single α_i random row effect is inclued, then the log-likelihood is calculated by integrating over this,

$$\log(f) = \sum_{i=1}^{n} \log(\int \prod_{j=1}^{p} \{f(y_{ij}|\beta_{0j}, \alpha_i, \ldots)\} f(\alpha_i) d\alpha_i),$$

where $f(\alpha_i)$ is the random effects distribution with mean zero and standard deviation given by the `row.params`. The integration is performed using standard Monte Carlo integration. This naturally extends to multiple random row effects structures.

Value

A list with the following components:

`logLik`	Value of the log-likelihood
`logLik.comp`	A vector of the log-likelihood values for each row of the response matrix, such that `sum(logLik.comp) = logLik`.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

`calc.marglogLik` for calculation of the log-likelihood marginalizing over one or more latent variables, and `calc.condlogLik` for calculation of the conditional log-likelihood for models where everything is treated as "fixed effects", including latent variables, row effects, and so on.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

## Example 1 - NULL model with site effects only
spiderfit_nb <- boral(y, family = "negative.binomial", row.eff = "fixed",
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
site_coef <- list(ID1 = apply(fit_mcmc[,grep("row.coefs.ID1", mcmc_names)],
    2,median))

## Calculate the log-likelihood at the posterior median
calc.logLik.lv0(y, family = "negative.binomial",
    lv.coefs =  coef_mat, row.eff = "fixed", row.params = site_coef)


## Example 2 - Model with environmental covariates and random row effects
X <- scale(spider$x)
spiderfit_nb2 <- boral(y, X = X, family = "negative.binomial", row.eff = "random",
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb2)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
X_coef_mat <- matrix(apply(fit_mcmc[,grep("X.coefs",mcmc_names)],
    2,median),nrow=p)
site.sigma <- list(ID1 =
```

```
    median(fit_mcmc[,grep("row.sigma.ID1", mcmc_names)]))


## Calculate the log-likelihood at the posterior median
calc.logLik.lv0(y, X = spider$x, family = "negative.binomial",
    row.eff = "random",lv.coefs =  coef_mat, X.coefs = X_coef_mat,
    row.params = site.sigma)

## End(Not run)
```

`calc.marglogLik` *Marginal log-likelihood for a fitted model*

Description

[Deprecated]

Calculates the marginal log-likelihood for a set of parameter estimates from a fitted model, whereby the latent variables and random effects (if applicable) are integrated out. The integration is performed using Monte Carlo integration. WARNING: As of version 1.9, this function is no longer being maintained (and probably does not work properly, if at all)!

Usage

```
calc.marglogLik(y, X = NULL, family, trial.size = 1, lv.coefs,
     X.coefs = NULL, row.eff = "none", row.params = NULL,
     row.ids = NULL,offset = NULL, num.lv, lv.mc = NULL,
     cutoffs = NULL, powerparam = NULL)
```

Arguments

`y`	The response matrix that the model was fitted to.
`X`	The covariate matrix used in the model. Defaults to `NULL`, in which case it is assumed no model matrix was used.
`family`	Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression). Please see `about.distributions` for information on distributions available in boral overall.
`trial.size`	Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

lv.coefs	The response-specific intercept, coefficient estimates relating to the latent variables, and dispersion parameters from the fitted model.
X.coefs	The coefficients estimates relating to the covariate matrix from the fitted model. Defaults to NULL, in which it is assumed there are no covariates in the model.
row.eff	Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and standard deviation given by row.params. Defaults to "none".
row.params	Parameters corresponding to the row effect from the fitted model. If row.eff = "fixed", then these are the fixed effects and should have length equal to the number of columns in the response matrix. If row.eff = "random", then this is standard deviation for the random effects normal distribution. If row.eff = "none", then this argument is ignored.
row.ids	A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see boral for details. Defaults to NULL, so that if row.params = NULL then the argument is ignored, otherwise if row.params is supplied then row.ids = matrix(1:nrow(y), ncol = 1) i.e., a single, row effect unique to each row. An internal check is done to see row.params and row.ids are consistent in terms of arguments supplied.
offset	A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to NULL.
num.lv	The number of latent variables used in the fitted model. For models with no latent variables, please use calc.logLik.lv0 to calculate the log-likelihood.
lv.mc	A matrix used for performing the Monte Carlo integration. Defaults to NULL, in which case a matrix is generated within the function.
cutoffs	Common cutoff estimates from the fitted model when any of the columns of the response matrix are ordinal responses. Defaults to NULL.
powerparam	Common power parameter from the fitted model when any of the columns of the response matrix are tweedie responses. Defaults to NULL.

Details

For an nxp response matrix $\boldsymbol{Y}$, suppose we fit a model with one or more latent variables. If we denote the latent variables by $\boldsymbol{u}_i; i = 1, \ldots, n$, then the marginal log-likelihood is given by

$$\log(f) = \sum_{i=1}^{n} \log(\int \prod_{j=1}^{p} \{ f(y_{ij}|\boldsymbol{u}_i, \beta_{0j}, \boldsymbol{\theta}_j, \ldots) \} f(\boldsymbol{u}_i) d\boldsymbol{u}_i),$$

where $f(y_{ij}|\cdot)$ is the assumed distribution for column j, β_{0j} are the response-specific intercepts, $\boldsymbol{\theta}_j$ are the response-specific latent variable coefficients, and ... generically denotes anything else

included in the model, e.g. row effects, dispersion parameters etc... The quantity $f(\boldsymbol{u}_i)$ denotes the distribution of the latent variable, which is assumed to be standard multivariate Gaussian. Standard Monte Carlo integration is used for calculating the marginal likelihood. If `lv.mc = NULL`, the function automatically generates a matrix as
`lv.mc <- rmvnorm(1000, rep(0,num.lv))`. If there is a need to apply this function numerous times, we recommend a matrix be inserted into `lv.mc` to speed up computation.

The key difference between this and the conditional likelihood (see `calc.condlogLik`) is that the marginal likelihood treats the latent variables as "random effects" and integrates over them, whereas the conditional likelihood treats the latent variables as "fixed effects".

Please note the function is written conditional on all regression coefficients. Therefore, if traits are included in the model, in which case the regression coefficients $\beta_{0j}, \boldsymbol{\beta}_j$ become random effects instead (please see `about.traits`), then the calculation of the log-likelihood does NOT take this into account, i.e. does not marginalize over them! Likewise if more than two columns are ordinal responses, then the regression coefficients β_{0j} corresponding to these columns become random effects, and the calculation of the log-likelihood also does NOT take this into account, i.e. does not marginalize over them!

When a single α_i random row effect is inclued, then the log-likelihood is calculated by integrating over this,

$$\log(f) = \sum_{i=1}^{n} \log(\int \prod_{j=1}^{p} \{ f(y_{ij}|\boldsymbol{u}_i, \beta_{0j}, \alpha_i, \ldots) \} f(\boldsymbol{u}_i) f(\alpha_i) d\alpha_i),$$

where $f(\alpha_i)$ is the random effects distribution with mean zero and standard deviation given by the `row.params`. The integration is again performed using standard Monte Carlo integration. This naturally extends to multiple random row effects structures.

Value

A list with the following components:

`logLik`	Value of the marginal log-likelihood.
`logLik.comp`	A vector of the log-likelihood values for each row of the response matrix, such that `sum(logLik.comp) = logLik`.

Warnings

As of version 1.6, this function is longer updated!

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

`calc.condlogLik` for calculation of the conditional log-likelihood; `calc.logLik.lv0` to calculate the conditional/marginal log-likelihood for a model with no latent variables.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

## Example 1 - model with two latent variables, site effects,
##  and no environmental covariates
spiderfit_nb <- boral(y, family = "negative.binomial",
    lv.control = list(num.lv = 2), row.eff = "fixed", save.model = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
site_coef <- list(ID1 = apply(fit_mcmc[,grep("row.coefs.ID1", mcmc_names)],
    2,median))

## Calculate the marginal log-likelihood at the posterior median
calc.marglogLik(y, family = "negative.binomial",
    lv.coefs = coef_mat, row.eff = "fixed", row.params = site_coef,
    num.lv = 2)


## Example 2 - model with one latent variable, no site effects,
##  and environmental covariates
spiderfit_nb2 <- boral(y, X = spider$x, family = "negative.binomial",
     lv.control = list(num.lv = 2), save.model = TRUE,
     mcmc.control = example_mcmc_control, model.name = testpath)

## Extract all MCMC samples
fit_mcmc <- get.mcmcsamples(spiderfit_nb2)
mcmc_names <- colnames(fit_mcmc)

## Find the posterior medians
coef_mat <- matrix(apply(fit_mcmc[,grep("lv.coefs",mcmc_names)],
    2,median),nrow=p)
X_coef_mat <- matrix(apply(fit_mcmc[,grep("X.coefs",mcmc_names)],
```

```
    2,median),nrow=p)

## Calculate the log-likelihood at the posterior median
calc.marglogLik(y, X = spider$x, family = "negative.binomial",
    lv.coefs = coef_mat, X.coefs = X_coef_mat, num.lv = 2)

## End(Not run)
```

`calc.varpart` *Variance partitioning for a latent variable model*

Description

[Stable]

For each response (species), partition the variance of the linear predictor into components associated with (groups of) the covariates, the latent variables, and any row effects and response-specific random intercepts. If traits are also included in the model, then it also calculates an R-squared value for the proportion of the variance in the environmental response (due to the covariates) which can be explained by traits.

Usage

```
calc.varpart(object, groupX = NULL)
```

Arguments

`object`	An object of class "boral".
`groupX`	A vector of group indicator variables, which allows the variance partitioning to be done for groups of covariates (including the intercept) i.e., how much of the total variation does a certain subset of the covariates explain. Defaults to `NULL`, in whih case all the covariates are treated as single group.

Details

As an alternative to looking at differences in trace of the residual covariance matrix (Hui et al., 2014; Warton et al., 2015), an alternative way to quantify the amount of variance explained by covariates, traits, row effects, response-specific random intercepts, is to perform a variance decomposition of the linear predictor of a latent variable model (Ovaskainen et al., 2017). In particular, for a general model the linear predictor for response $j = 1, \ldots, p$ at row $i = 1, \ldots, n$ is given by

$$\eta_{ij} = \alpha_i + \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j + \boldsymbol{u}_i^\top \boldsymbol{\theta}_j,$$

where $\beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j$ is the component of the linear predictor due to the covariates $\boldsymbol{X}$ plus an intercept, $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ is the component due to response-specific random intercept, $\boldsymbol{u}_i^\top \boldsymbol{\theta}_j$ is the component due to the latent variables, and α_i is the component due to one or more fixed or random row effects. Not all of these components may be included in the model, and the above is just representing the general

case. The regression coefficients $\boldsymbol{\beta}_j$ may be further as random effects and regressed against traits; please see `about.traits` for further information on this.

For the response, a variation partitioning of the linear predictor is performed by calculating the variance due to the components in η_{ij} and then rescaling them to ensure that they sum to one. The general details of this type of variation partitioning is given in Ovaskainen et al., (2017); see also Nakagawa and Schielzeth (2013) for R-squared and proportion of variance explained in the case of generalized linear mixed model. In brief, for response $j = 1, \ldots, p$:

- the variance due to the covariates and intercept is given by the variance of $\beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j$ calculated across the n rows;
- the variance due to (all) the response-respecific random intercepts is given by the (sum of the) variances for each of the elements of b_j
- the variance due to (all) the random row effects is given by variance of α_i calculated across the n rows for fixed row effects (`row.eff = "fixed"`), and given by the (sum of the) variance σ_α^2 for random row effects (`row.eff = "random"`);
- the variance due the latent variables is given by the diagonal elements of $\boldsymbol{\theta}_j^\top \boldsymbol{\theta}_j$.

After scaling, we can then obtain the proportion of variance for each response which is explained by the variance components. These proportions are calculated for each MCMC sample and then average acrossed them to calculate a posterior mean variance partitioning.

If `groupX` is supplied, the variance due to the covariates is done based on subsets of the covariates (including the intercept) as identified by `groupX`, and then rescaled correspondingly. This is useful if one was to, for example, quantify the proportion of variation in each response which is explained by each covariate.

If a fitted model also containing traits, which are included to help explain/mediate differences in species environmental responses, then the function calculates R^2 value for the proportion of variance in the covariates which is explained by the traits. In brief, this is calculated based the correlation between $\beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j$ and $\tau_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\tau}_j$, where τ_{0j} and $\boldsymbol{\tau}_j$ are the "predicted" values of the species coefficients based on values i.e., $\tau_{0j} = \kappa_{01} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_1$ and $\tau_{jk} = \kappa_{0k} + \boldsymbol{traits}_j^\top \boldsymbol{\kappa}_k$ for element k in $\boldsymbol{\tau}_j$.

Value

A list containing the following components, if applicable:

`varpart.X`	Vector containing the proportion of variance (in the linear predictor) for each response, which is explained by the covariate matrix.
`varpart.lv`	Vector containing the proportion of variance (in the linear predictor) for each response, which is explained by the latent variables.
`varpart.row`	Vector containing the proportion of variance (in the linear predictor) for each response, which is explained by the row effects.
`varpart.ranef`	Vector containing the proportion of variance (in the linear predictor) for each response, which is explained by the response-specific random intercepts.
`R2.traits`	Vector containing the proportion of variance due to the covariates for each response, which can be explained by traits for each response.

Warnings

There is considerable controversy over exactly what quantities such as R-squared and proportion of variance explained are in the case mixed models and latent variable models, and how they can interpreted e.g., what is considered a high value for the proportion of variance by the covariates, is it consistent with whether the coefficients are significantly different from zero or not; see for instance R2 controversy.

When reporting these values, researchers should be at least aware of this and that there are multiple ways of manufacturing such quantities, with no single best approach e.g., using relative changes in trace of the residual covariance matrix, relative changes in marginal and conditional log-likelihoods are other possible approaches.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Nakagawa, S., and Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution 4, 133-142.
- Ovaskainen et al. (2017). How to make more out of community data? A conceptual framework and its implementation as models and software. Ecology Letters 20, 561-576.
- Hui et al. (2014). Model-based approaches to unconstrained ordination. Methods in Ecology and Evolution, 6, 399-411.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, 30, 766-779.

Examples

```
## Not run:
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1 - model with X variables, two latent variables, and no row effects
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
     lv.control = list(num.lv = 2),
     save.model = TRUE, mcmc.control = example_mcmc_control,
```

```
     model.name = testpath)

## Partition variance for each species into that explained by covariates
## and by the latent variables
dovar <- calc.varpart(spiderfit_nb)

## Consider the intercept and first two covariates in X as one group,
## and remaining four covariates in X as another group,
## then partition variance for each species based on these groups.
dovar <- calc.varpart(spiderfit_nb, groupX = c(1,1,1,2,2,2,2))


## Example 1b - model with X variables, two latent variables, and
## species-specific random intercepts at a so-called region level
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2),
    ranef.ids = data.frame(subregion = rep(1:7,each=4)),
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Partition variance for each species into that explained by covariates
## and by the latent variables
dovar <- calc.varpart(spiderfit_nb)

## Consider the intercept and first two covariates in X as one group,
## and remaining four covariates in X as another group,
## then partition variance for each species based on these groups.
dovar <- calc.varpart(spiderfit_nb, groupX = c(1,1,1,2,2,2,2))


## Example 2 - model fitted to count data, no site effects, and
## two latent variables, plus traits included to explain environmental responses
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)
## Just for fun, the regression coefficients for the second column of X,
## corresponding to the third element in the list example_which_traits,
## will be estimated separately and not regressed against traits.
example_which_traits[[3]] <- 0

fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", mcmc.control = example_mcmc_control,
    save.model = TRUE, model.name = testpath)

## Partition variance for each species due to covariates in X
## and latent variables. Also calculate proportion of variance
## due to the covariates which can be explained by traits
dovar <- calc.varpart(fit_traits)
```

```
## End(Not run)
```

`coefsplot` *Caterpillar plots of the regression coefficients from a fitted model*

Description

[Stable]

Constructs horizontal line plot (point estimate and HPD intervals), otherwise known as "caterpillar plots", for the response-specific regression coefficients corresponding to the covariate in the fitted model. If a fourth-corner model is fitted, then "caterpillar plots" can optionally be produced for all the fourth-corner regression coefficients.

Usage

```
coefsplot(covname, object, fourthcorner = FALSE, labely = NULL, est = "median", ...)
```

Arguments

`covname`	The name of one of the covariates in the fitted model. That is, it must be a character vector corresponding to one of the elements in `colnames(object$X.coefs.median)`. If `fourthcorner = TRUE`, then this argument is ignored.
`object`	An object for class "boral".
`fourthcorner`	If set to `TRUE`, then a caterpillar plot of the fourth-corner regression coefficients, as given by `object$traits.coefs.median` say, are plotted instead, assuming a model involving traits is fitted. Defaults to `FALSE`, in which case the plot is of response-specific regression coefficients. If `fourthcorner = TRUE`, then both the `covname` and `labely` arguments are ignored.
`labely`	Controls the labels on the y-axis for the line plot. If it is not `NULL`, then it must be a vector either of length 1 or the same length as the number of columns in the response matrix in the fitted boral object. In the former, it is treated as the y-axis label. In the latter, it is used in place of the column names of the response matrix to label each line. Defaults to `NULL`, in which the each line in the plot is labeled according to the columns of the response matrix, or equivalently `rownames(object$X.coefs.median)`. If `fourthcorner = TRUE`, then this argument is ignored.
`est`	A choice of either the posterior median (`est = "median"`) or posterior mean (`est = "mean"`), which are then used as the point estimates in the lines. Default is posterior median.
`...`	Additional graphical options to be included in. These include values for `cex`, `cex.lab`, `cex.axis`, `cex.main`, `lwd`, and so on.

Details

For each response (column of the response matrix, the horizontal line or "caterpillar" is constructed by first marking the point estimate (posterior mean or median) with an "x" symbol. Then the line is construed based on the lower and upper limits of the highest posterior density (HPD) intervals as found in `object$hpdintervals`. By default, these are 95% HPD intervals. To complete the plot, a vertical dotted line is drawn to denote the zero value. All HPD intervals that include zero are colored gray, while HPD intervals that exclude zero are colored black.

For plots of fourth-corner regression coefficients, the coefficients are labelled such that on the left vertical axis the names of the covariates includes in the model are given, while on the right vertical axis the names of traits included in the model are given. Please see the `about.traits` for more about fourth-corner models where traits are included to help explain differences in species environmental responses to covariates.

The graph is probably better explained by, well, plotting it using the toy example below! Thanks to Robert O'Hara for suggesting and providing the original code for this function.

Value

If SSVS was applied individually to each coefficient of the covariate matrix when fitting the model, then the posterior probabilities of including the specified covariate are printed out i.e.,
those from `object$ssvs.indcoefs.mean`.

For fourth-corher models, if SSVS was applied individually to fourth-corner coefficients when fitting the model, then the posterior probabilities of including the specified coefficient are printed out i.e.,
those from `object$ssvs.traitscoefs.mean`.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

`ranefsplot` for horizontal line or "caterpillar plot" of the response-specific random effects predictons (if applicable), `caterplot` from the `mcmcplots` package, as well as the `ggpubr` package, for other, sexier caterpillar plots.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
```

```
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

X <- scale(spider$x)
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2), mcmc.control = example_mcmc_control,
    model.name = testpath)


## Do separate line plots for all the coefficients of X
par(mfrow=c(2,3), mar = c(5,6,1,1))
sapply(colnames(spiderfit_nb$X), coefsplot,
    spiderfit_nb)


## Consider a model based on Example 5a in the main boral help file
## The model is fitted to count data, no site effects, two latent variables,
## plus traits included to explain environmental responses
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)
## Just for fun, the regression coefficients for the second column of X,
## corresponding to the third element in the list example_which_traits,
## will be estimated separately and not regressed against traits.
example_which_traits[[3]] <- 0

fit_traits <- boral(y, X = X, traits = traits,
                    which.traits = example_which_traits, family = "negative.binomial",
                    mcmc.control = example_mcmc_control, model.name = testpath,
                    save.model = TRUE)

summary(fit_traits)

par(mar = c(3,10,2,10))
coefsplot(object = fit_traits, fourthcorner = TRUE)

## End(Not run)
```

`create.life` **[Stable]** *Simulate a Multivariate response matrix*

Description

Simulate a multivariate response matrix, given parameters such as but not necessarily all of: family, number of latent variables and related coefficients, an matrix of explanatory variables and related coefficients, row effects, response-specific random intercepts, cutoffs for cumulative probit regression of ordinal responses, and so on.

Usage

```
create.life(true.lv = NULL, lv.coefs,
     lv.control = list(num.lv = 0, type = "independent",
     lv.covparams = NULL, distmat = NULL),
     X = NULL, X.coefs = NULL,
     traits = NULL, traits.coefs = NULL, family,
     row.eff = "none", row.params = NULL, row.ids = NULL,
     true.ranef = NULL, ranef.params = NULL, ranef.ids = NULL,
     offset = NULL, trial.size = 1, cutoffs = NULL, powerparam = NULL,
     manual.dim = NULL, save.params = FALSE)


## S3 method for class 'boral'
simulate(object, nsim = 1, seed = NULL, new.lvs = FALSE, new.ranefs = FALSE,
     distmat = NULL, est = "median", ...)
```

Arguments

object	An object of class "boral".
nsim	Number of multivariate response matrices to simulate. Defaults to 1.
seed	Seed for dataset simulation. Defaults to NULL, in which case no seed is set.
new.lvs	If FALSE, then true latent variables are obtained from object. If TRUE, then new true latent variables are generated.
new.ranefs	If FALSE, then true response-specific random intercepts are obtained from object. If TRUE, then new random intercepts are generated.
distmat	A distance matrix required to calculate correlations across sites when a non-independent correlation structure on the latent variables is imposed, when new.lvs = TRUE and object$lv.type != "independent".
est	A choice of either the posterior median (est = "median") or posterior mean (est = "mean"), which are then treated as estimates and the fitted values are calculated from. Default is posterior median.
true.lv	A matrix of true latent variables. With multivariate abundance data in ecology for instance, each row corresponds to the true site ordination coordinates. If supplied, then simulation is based of these true latent variables. If NULL, then the function looks to the argument lv.control to see what to do. Defaults to NULL.
lv.coefs	A matrix containing response-specific intercepts, latent variable coefficients relating to true.lv, and dispersion parameters.

`lv.control` This argument is utilized if `true.lv = NULL`, in which case the function uses this argument to determine how to simulate new, true latent variables. A list (currently) with the following arguments:

- *num.lv:* which specifies the number of true latent variables to generate. Defaults to 0.
- *type:* which specifies the type the correlation structure of the latent variables (across sites). Defaults to independence correlation structure.
- *lv.covparams:* which is a vector containing one or two elements required if parameterizing a non-independence spatial correlation structure of the latent variables.
- *distmat:* which a distance matrix required to calculate correlations across sites when a non-independence correlation structure on the latent variables is imposed.

Please see `about.lvs` for more information.

`X` A model matrix of covariates (otherwise known as the covariate matrix), which can be included as part of the data generation. Defaults to `NULL`, in which case no model matrix is used. No intercept column should be included.

`X.coefs` The coefficients relating to the covariate matrix. Defaults to `NULL`. This argument needs to be supplied if the covariate matrix is supplied and no trait matrix is supplied.

`traits` A model matrix of species traits (otherwise known as the covariate matrix), which can be included as part of the model. Defaults to `NULL`, in which case no matrix was used. No intercept column should be included in the trait matrix, as it is included automatically.

`traits.coefs` A matrix of coefficients that are used to generate "new" response-specific intercepts and `X.coefs`. The number of rows should equal to `(ncol(X)+1)` and the number of columns should equal to `(ncol(traits)+2)`.

How this argument works is as follows: when both `traits` and `traits.coefs` are supplied, then new response-specific intercepts (i.e. the first column of `lv.coefs` is overwritten) are generated by simulating from a normal distribution with mean equal to `crossprod(c(1,traits), traits.coefs[1,1:(ncol(traits.coefs)-1)])` and standard deviation `traits.coefs[1,ncol(traits.coefs)]`. In other words, the last column of `trait.coefs` provides the standard deviation of the normal distribution, with the other columns being the regression coefficients in the mean of the normal distribution. Analogously, new `X.coefs` are generated in the same manner using the remaining rows of `trait.coefs`. Please see `about.traits` for more information.

It is important that highlight then with in this data generation mechanism, the new response-specific intercepts and `X.coefs` are now random effects, being drawn from a normal distribution.

Defaults to `NULL`, in conjuction with `traits = NULL`.

`family` Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distribu-

tions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link).

Please see `about.distributions` for information on distributions available in boral overall.

`row.eff` Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and standard deviation given by `row.params`. Defaults to "none".

`row.params` Parameters corresponding to the row effects. If `row.eff = "fixed"`, then these are the fixed effects and should have length equal to the number of columns in the response matrix. If `row.eff = "random"`, then this is the standard deviation for the random effects normal distribution. If `row.eff = "none"`, then this argument is ignored.

`row.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see `boral` for details. Defaults to `NULL`, so that if `row.params = NULL` then the argument is ignored, otherwise if `row.params` is supplied then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row.

`true.ranef` A list of true response-specific random intercepts. If supplied, it should be a list of length `length(ranef.ids)`, where the k-th element is the matrix a matrix where the number of rows is equal to the number of responses, and the number of columns is equal to `length(unique(ranef.ids[,k]))`. If `NULL`, then the function looks to the arguments `ranef.ids` and/or `ranef.params` to see what to do. Defaults to `NULL`.

`ranef.params` Parameters corresponding to standard deviation for the the response-specific random intercepts distribution. If supplied, it should be a matrix, where the number of rows is equal to the number of responses, and the number of columns equal to `length(ranef.ids)`. If `NULL`, then the function looks to the arguments `ranef.ids` and/or `true.ranef` to see what to do. Defaults to `NULL`.

`ranef.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j; please see `about.ranefs` for details. Defaults to `NULL`, in which case it is assumed no random intercepts are to be included in the model. If supplied, then either one of `true.ranef` or `ranef.params` must also be supplied. If `true.ranef` is supplied, then these are

used as the true random intercepts; if ranef.params is supplied, then response-specific random intercepts are generated from a normal distribution with mean zero and (response-specific) standard deviation based on the elements of ranef.params.

offset — A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to NULL.

trial.size — Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

cutoffs — A vector of common common cutoffs for proportional odds regression when any of family is ordinal. They should be increasing order. Defaults to NULL.

powerparam — A common power parameter for tweedie regression when any of family is tweedie. Defaults to NULL.

manual.dim — A vector of length 2, containing the number of rows (n) and columns (p) for the multivariate response matrix. This is a "backup" argument only required when create.life can not determine how many rows or columns the multivariate response matrix should be.

save.params — If save.params = TRUE, then all parameters provided as input and/or generated are returned, in addition to the simulated multivariate response matrix. Defaults to FALSE.

... — Not used.

Details

create.life gives the user flexibility to control the true parameters of the model from which the multivariate responses matrices are generated from. For example, if true.lv is supplied, then the data generation mechanism is based on this set of true latent variables. If true.lv = NULL, then the function looks to lv.control to determine whether and how the true latent variables are to be simulated.

simulate makes use of the generic function of the same name in R: it takes a fitted model, treats either the posterior medians and mean estimates from the model as the true parameters, and generates response matrices based off that. There is control as to whether the latent variables and/or response-specific random intercepts obtained from the fitted model are used, or new ones are generated.

Value

If create.life is used, then: 1) if save.params = FALSE, a n by p multivariate response matrix is returned only, 2) if save.params = TRUE, then a list containing the element resp which is a n times p multivariate response matrix, as well as other elements for the parameters used in the true model are returned.

If simulate is used, then a three dimensional array of dimension n by p by nsim is returned. The same latent variables can be used each time (new.lvs = FALSE), or new true latent variables can be generated each time (new.lvs = TRUE).

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]
Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

boral for the default function for model fitting.

Examples

```
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1a - Simulate a response matrix of normally distributed data
library(mvtnorm)

## 30 rows (sites) with two latent variables
true_lv <- rbind(rmvnorm(n=15,mean=c(1,2)),rmvnorm(n=15,mean=c(-3,-1)))
## 30 columns (species)
true_lv_coefs <- cbind(matrix(runif(30*3),30,3),1)

X <- matrix(rnorm(30*4),30,4)
## 4 explanatory variables
X.coefs <- matrix(rnorm(30*4),30,4)

simy <- create.life(true.lv = true_lv, lv.coefs = true_lv_coefs,
    X = X, X.coefs = X.coefs, family = "normal")

## Not run:
fit_simdata <- boral(simy, X = X, family = "normal", lv.control = list(num.lv = 2),
    mcmc.control = example_mcmc_control, model.name = testpath)

summary(fit_simdata)

## End(Not run)

## Example 1b - Include a nested random row effect
## 30 subregions nested within six regions
example_row_ids <- cbind(1:30, rep(1:6,each=5))
## Subregion has a small std deviation; region has a larger one
true_ranef_sigma <- list(ID1 = 0.5, ID2 = 2)

simy <- create.life(true.lv = true_lv, lv.coefs = true_lv_coefs,
    X = X, X.coefs = X.coefs, row.eff = "random",
    row.params = true_ranef_sigma, row.ids = example_row_ids, family = "normal",
    save.params = TRUE)
```

```
## Example 1c - Same as example 1b except new, true latent variables are generated
simy <- create.life(true.lv = NULL, lv.coefs = true_lv_coefs,
    X = X, X.coefs = X.coefs, row.eff = "random",
    row.params = true_ranef_sigma, row.ids = example_row_ids, family = "normal",
    save.params = TRUE)


## Example 1d - Same as example 1a except new, true latent variables are generated
##   with a non-independent correlation structure using a fake distance matrix
makedistmat <- as.matrix(dist(1:30))
simy <- create.life(true.lv = NULL, lv.coefs = true_lv_coefs,
    lv.control = list(num.lv = 2, type = "exponential", lv.covparams = 5,
          distmat = makedistmat),
    X = X, X.coefs = X.coefs, row.eff = "random",
    row.params = true_ranef_sigma, row.ids = example_row_ids, family = "normal",
    save.params = TRUE)


## Example 1e - Similar to 1b, except instead of a nested random row effect,
## it includes a species-specific random interept at the region level
example_ranef_ids <- data.frame(region = rep(1:6,each=5))
## Subregion has a small std deviation; region has a larger one
true_ranef_sigma <- matrix(runif(nrow(true_lv_coefs)),
     nrow = nrow(true_lv_coefs), ncol = 1)

simy <- create.life(true.lv = true_lv, lv.coefs = true_lv_coefs,
    X = X, X.coefs = X.coefs, ranef.ids = example_ranef_ids,
    ranef.params = true_ranef_sigma, family = "normal",
    save.params = TRUE)


## Example 2 - Simulate a response matrix of ordinal data
## 30 rows (sites) with two latent variables
true_lv <- rbind(rmvnorm(15,mean=c(-2,-2)),rmvnorm(15,mean=c(2,2)))
## 10 columns (species)
true_lv_coefs <- rmvnorm(10,mean = rep(0,3));
## Cutoffs for proportional odds regression (must be in increasing order)
true.ordinal.cutoffs <- seq(-2,10,length=10-1)

simy <- create.life(true.lv = true_lv, lv.coefs = true_lv_coefs,
     family = "ordinal", cutoffs = true.ordinal.cutoffs, save.params = TRUE)

## Not run:
fit_simdata <- boral(y = simy$resp, family = "ordinal", lv.control = list(num.lv = 2),
      mcmc.control = example_mcmc_control, model.name = testpath)

## End(Not run)

## Not run:
## Example 3 - Simulate a response matrix of count data based off
## a fitted model involving traits (ants data from mvabund)
library(mvabund)
```

```
data(antTraits)

y <- antTraits$abun
X <- as.matrix(antTraits$env)
## Include only traits 1, 2, and 5, plus an intercept
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
## Please see help file for boral regarding the use of which.traits
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)

fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", lv.control = list(num.lv = 2),
     mcmc.control = example_mcmc_control, model.name = testpath)

## The hard way
simy <- create.life(true.lv = fit_traits$lv.mean,
     lv.coefs = fit_traits$lv.coefs.median, X = X,
     X.coefs = fit_traits$X.coefs.median, traits = traits,
     traits.coefs = fit_traits$traits.coefs.median, family = "negative.binomial")

## The easy way, using the same latent variables as the fitted model
simy <- simulate(object = fit_traits)

## The easy way, generating new latent variables
simy <- simulate(object = fit_traits, new.lvs = TRUE)


## Example 4 - simulate Bernoulli data, based on a model with two latent variables,
## no site variables, with two traits and one environmental covariates
## This example is a proof of concept that traits can used
## to explain environmental responses
library(mvtnorm)

n <- 100; s <- 50
X <- as.matrix(scale(1:n))
colnames(X) <- c("elevation")

traits <- cbind(rbinom(s,1,0.5), rnorm(s))
## one categorical and one continuous variable
colnames(traits) <- c("thorns-dummy","SLA")

simfit <- list(lv.coefs = cbind(rnorm(s), rmvnorm(s, mean = rep(0,2))),
     lv.control = list(num.lv = 2, type = "independent"),
    traits.coefs = matrix(c(0.1,1,-0.5,1,0.5,0,-1,1), 2, byrow = TRUE))
rownames(simfit$traits.coefs) <- c("beta0","elevation")
colnames(simfit$traits.coefs) <- c("kappa0","thorns-dummy","SLA","sigma")

simy <- create.life(lv.control = simfit$lv.control, lv.coefs = simfit$lv.coefs,
    X = X, traits = traits, traits.coefs = simfit$traits.coefs,
    family = "binomial")
```

```
example_which_traits <- vector("list",ncol(X)+1);
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)
fit_traits <- boral(y = simy, X = X, traits = traits,
    which.traits = example_which_traits, family = "binomial",
    lv.control = list(num.lv = 2), save.model = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)


## Example 5 -- extend Example 2e in the main boral help file to simulate data from a
##   fitted model
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)

spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

simulate(object = spiderfit_nb)

simulate(object = spiderfit_nb, new.ranefs = TRUE)

## End(Not run)
```

`ds.residuals` *Dunn-Smyth Residuals for a fitted model*

Description

[Stable]

Calculates the Dunn-Smyth residuals for a fitted model and, if some of the responses are ordinal, a confusion matrix between predicted and true levels.

Usage

```
ds.residuals(object, est = "median", include.ranef = TRUE)
```

Arguments

`object`	An object for class "boral".
`est`	A choice of either the posterior median (`est = "median"`) or posterior mean (`est = "mean"`), which are then treated as parameter estimates and the residuals are calculated from. Default is posterior median.
`include.ranef`	If response-specific random intercepts were included as part of the fitted model, then this determines whether the predicted random effects will be used in the calculated of the fitted values and thus residuals. When set to `TRUE`, which is the

default, then they are included (using either the posterior mean and posterior median predictor). When set to FALSE, they are not included. The former leads to what are sometimes called conditional residuals, while the latter are sometimes called marginal residuals.

Details

Details regarding Dunn-Smyth residuals, based on the randomized quantile residuals of Dunn and Smyth (1996), can be found in `plot.manyglm` function in the `mvabund` package (Wang et al., 2012) where they are implemented in all their glory. Due their inherent stochasticity, Dunn-Smyth residuals will be slightly different each time this function is run. As with other types of residuals, Dunn-Smyth residuals can be used in the context of residual analysis.

For ordinal responses, a single confusion matrix between the predicted levels (as based on the class with the highest probability) and true levels is aso returned. The table pools the results over all columns assumed to be ordinal.

The Dunn-Smyth residuals are calculated based on a point estimate of the parameters, as determined by the argument `est`. A fully Bayesian approach would calculate the residuals by averaging over the posterior distribution of the parameters i.e., ergodically average over the MCMC samples. In general however, the results (as in the trends seen in residual analysis) from either approach should be very similar.

Check out also the awesome `DHARMa` package for calculation of Dunn-Smyth and probability integral transform residuals in other regression models.

Value

A list containing `agree.ordinal` which is a single confusion matrix for ordinal columns, and `residuals` which contains Dunn-Smyth residuals.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Dunn, P. K., and Smyth, G. K. (1996). Randomized quantile residuals. Journal of Computational and Graphical Statistics, 5, 236-244.
- Wang et al. (2012). mvabund-an R package for model-based analysis of multivariate abundance data. Methods in Ecology and Evolution, 3, 471-474.

See Also

`plot.boral` for constructing residual analysis plots directly; `fitted.boral` which calculated fitted values from a model.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun

spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
     row.eff = "fixed", mcmc.control = example_mcmc_control, model.name = testpath)

ds.residuals(spiderfit_nb)

## End(Not run)
```

`fitted.boral` *Extract Model Fitted Values for an boral object*

Description

[Stable]

Calculated the fitted values based on the response or linear predictor scale, by using the posterior medians or means of the parameters.

Usage

```
## S3 method for class 'boral'
fitted(object, est = "median", include.ranef = TRUE, linear.predictor = FALSE, ...)
```

Arguments

`object`	An object of class "boral".
`est`	A choice of either the posterior median (`est = "median"`) or posterior mean (`est = "mean"`), which are then treated as estimates and the fitted values are calculated from. Default is posterior median.
`include.ranef`	If response-specific random intercepts were included as part of the fitted model, then this determines whether the predicted random effects will be included in the fitted values. When set to `TRUE`, which is the default, then they are included (using either the posterior mean and posterior median predictor). When set to `FALSE`, they are not included. The former are sometimes called conditional fitted values, while the latter are sometimes called marginal fitted values.

`linear.predictor`
: Determines the scale on which to return the fitted values. When set to `TRUE`, it returns the fitted values on the linear predictor scale. When set to `FALSE`, which is the defaul behavior, the fitted values are on the response scale. Note things are slightly more complicated for zero truncated distributions because, the log-link connects the mean of the *untruncated* distribution to the linear predictor. Therefore if `linear.predictor = TRUE`, then the linear predictor is returned. But if `linear.predictor = FALSE`, then actual mean value is returned.

`...`
: Not used.

Details

This fitted values here are calculated based on a point estimate of the parameters, as determined by the argument `est`. A fully Bayesian approach would calculate the fitted values by averaging over the posterior distribution of the parameters i.e., ergodically average over the MCMC samples. For simplicity and speed though (to avoid generation of a large number of predicted values), this is not implemented.

Value

A list containing `ordinal.probs` which is an array with dimensions (number of rows of the response matrix) x (number of columns of the response matrix) x (no. of levels) containing the predicted probabilities for ordinal columns, and `out` which is a matrix of the same dimension as the original response matrix containing the fitted values. For ordinal columns, the "fitted values" are defined as the level/class that had the highest fitted probability.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

`plot.boral` which uses the fitted values calculated from this function to construct plots for residual analysis, `ds.residuals` for calculating the Dunn-Smyth residuals for a fitted model.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
```

```
spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
     row.eff = "fixed", mcmc.control = example_mcmc_control, model.name = testpath)

fitted(spiderfit_nb)

## End(Not run)
```

`get.dic` *Extract Deviance Information Criterion for a fitted model*

Description

[Defunct]

Calculates the Deviance Information Criterion (DIC) for a model fitted using JAGS. WARNING: As of version 1.6, this function is no longer maintained (and probably doesn't work properly, if at all)!

Usage

```
get.dic(jagsfit)
```

Arguments

`jagsfit`	The `jags.model` component of the output, from a model fitted using `boral` with `save.model = TRUE`.

Details

Details regarding the Deviance Information Criterion may be found in (Spiegelhalter et al., 2002; Ntzoufras, 2011; Gelman et al., 2013). The DIC here is based on the conditional log-likelihood i.e., the latent variables (and row effects if applicable) are treated as "fixed effects". A DIC based on the marginal likelihood is obtainable from `get.more.measures`, although this requires a much longer time to compute. For models with overdispered count data, conditional DIC may not perform as well as marginal DIC (Millar, 2009)

Value

DIC value for the jags model.

Note

This function and consequently the DIC value is automatically returned when a model is fitted using `boral` with `calc.ics = TRUE`.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Gelman et al. (2013). Bayesian data analysis. CRC press.
- Millar, R. B. (2009). Comparison of hierarchical Bayesian models for overdispersed count data using DIC and Bayes' factors. Biometrics, 65, 962-969.
- Ntzoufras, I. (2011). Bayesian modeling using WinBUGS (Vol. 698). John Wiley & Sons.
- Spiegelhalter et al. (2002). Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64, 583-639.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
     save.model = TRUE, calc.ics = TRUE, mcmc.control = example_mcmc_control,
     model.name = testpath)

spiderfit_nb$ics ## DIC returned as one of several information criteria.

## End(Not run)
```

`get.enviro.cor` *Extract covariances and correlations due to shared environmental responses*

Description

[Stable]

Calculates the correlation between columns of the response matrix, due to similarities in the response to explanatory variables i.e., shared environmental response.

Usage

```
get.enviro.cor(object, est = "median", prob = 0.95)
```

Arguments

`object`	An object for class "boral".
`est`	A choice of either the posterior median (`est = "median"`) or posterior mean (`est = "mean"`), which are then treated as estimates and the fitted values are calculated from. Default is posterior median.
`prob`	A numeric scalar in the interval (0,1) giving the target probability coverage of the intervals, by which to determine whether the correlations are "significant". Defaults to 0.95.

Details

In both independent response and correlated response models, where the each of the columns of the response matrix $\boldsymbol{Y}$ are fitted to a set of covariates, the covariance and thus between two columns j and j' due to similarities in their response to the model matrix is calculated based on the linear predictors $\boldsymbol{x}_i^\top \boldsymbol{\beta}_j$ and $\boldsymbol{x}_i^\top \boldsymbol{\beta}_{j'}$), where $\boldsymbol{\beta}_j$ are response-specific coefficients relating to the explanatory variables.

For multivariate abundance data, the correlation calculated by this function can be interpreted as the correlation attributable to similarities in the environmental response between species. Such correlation matrices are discussed and found in Ovaskainen et al., (2010), Pollock et al., 2014.

Please note this correlation calculation does not include any row effects or any response-specific random intercepts.

Value

A list with the following components:

`cor`, `cor.lower`, `cor.upper`	A set of $p \times p$ correlation matrices, containing either the posterior median or mean estimate plus lower and upper limits of the corresponding (100×`prob`) % HPD interval.
`sig.cor`	A $p \times p$ correlation matrix containing only the "significant" correlations whose (100×`prob`) % HPD interval does not contain zero. All non-significant correlations are set to zero.
`cov`	A $p \times p$ covariance matrix.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Ovaskainen et al. (2010). Modeling species co-occurrence by multivariate logistic regression generates new hypotheses on fungal interactions. Ecology, 91, 2514-2521.
- Pollock et al. (2014). Understanding co-occurrence by modelling species simultaneously with a Joint Species Distribution Model (JSDM). Methods in Ecology and Evolution, 5, 397-406.

See Also

get.residual.cor, which calculates the residual correlation matrix for models involving latent variables.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
library(corrplot) ## For plotting correlations
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
     save.model = TRUE, mcmc.control = example_mcmc_control,
     model.name = testpath)

enviro.cors <- get.enviro.cor(spiderfit_nb)

corrplot(enviro.cors$sig.cor, title = "Shared response correlations",
 type = "lower", diag = FALSE, mar = c(3,0.5,2,1), tl.srt = 45)

## End(Not run)
```

---

get.hpdintervals *Highest posterior density intervals for a fitted model*

Description

[Stable]

Calculates the lower and upper bounds of the highest posterior density intervals for parameters and latent variables in a fitted model.

Usage

```
get.hpdintervals(y, X = NULL, traits = NULL, row.ids = NULL, ranef.ids = NULL,
 fit.mcmc, lv.control, prob = 0.95, num.lv = NULL)
```

Arguments

y	The response matrix that the model was fitted to.
X	The covariate matrix used in the model. Defaults to NULL, in which case it is assumed no model matrix was used.
traits	The trait matrix used in the model. Defaults to NULL, in which case it is assumed no traits were included.
row.ids	A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see boral for details. Defaults to NULL, in which case iti assumed no random effects were included in the model.
ranef.ids	A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j; please see about.ranefs for details. Defaults to NULL, in which case it is assumed no random intercepts are to be included in the model. If supplied, then response-specific random intercepts are assumed to come from a normal distribution with mean zero and unknown (response-specific) standard deviation.
fit.mcmc	All MCMC samples for the fitted model. These can be extracted by fitting a model using boral with save.model = TRUE, and then applying get.mcmcsamples(fit).
lv.control	A list (currently) with the following arguments: • *num.lv:* which specifies the number of true latent variables to generate. Defaults to 0. • *type:* which specifies the type the correlation structure of the latent variables (across sites). Defaults to independence correlation structure. • *distmat:* which a distance matrix required to calculate correlations across sites when a non-independence correlation structure on the latent variables is imposed. Please see about.lvs for more information.
prob	A numeric scalar in the interval (0,1) giving the target probability coverage of the intervals. Defaults to 0.95.
num.lv	Old argument superceded by lv.control. Defaults to NULL and ignored.

Details

The function uses the HPDinterval function from the coda package to obtain the HPD intervals. See HPDinterval for details regarding the definition of the HPD interval. For interpreting the results, please check the dimension names of each of the components below to better ascertain what is being printed.

Value

A list containing the following components, where applicable:

`lv.coefs`	An array giving the lower and upper bounds of the HPD intervals for the response-specific intercepts, latent variable coefficients, and dispersion parameters if appropriate.
`lv`	An array giving the and upper bounds of the HPD intervals for the latent variables.
`lv.covparams`	A matrix giving the lower and upper bounds of the HPD intervals for the parameters characterizing the correlation structure of the latent variables when they are assumed to be non-independent across rows.
`row.coefs`	A list with each element being a matrix giving the lower and upper bounds of the HPD intervals for row effects. The number of elements in the list should equal the number of row effects included in the model i.e., `ncol(row.ids)`.
`row.sigma`	A list with each element being a vector giving the lower and upper bounds of the HPD interval for the standard deviation of the normal distribution for the row effects. The number of elements in the list should equal the number of row effects included in the model i.e., `ncol(row.ids)`.
`ranef.coefs`	A list with each element being a array giving the lower and upper bounds of the HPD intervals for response-specific random intercepts. The number of elements in the list should equal the number of row effects included in the model i.e., `ncol(ranef.ids)`.
`ranef.sigma`	An array giving the lower and upper bounds of the HPD interval for the standard deviation of the normal distribution for the response-specific random intercepts. The number of elements in the list should equal the number of row effects included in the model i.e., `ncol(row.ids)`.
`X.coefs`	An array giving the lower and upper bounds of the HPD intervals for coefficients relating to the covariate matrix.
`traits.coefs`	An array giving the lower and upper of the HPD intervals for coefficients and standard deviation relating to the traits matrix.
`cutoffs`	A matrix giving the lower and upper bounds of the HPD intervals for common cutoffs in proportional odds regression.
`powerparam`	A vector giving the lower and upper bounds of the HPD interval for common power parameter in tweedie regression.

Warnings

- HPD intervals tend to be quite wide, and inference is somewhat tricky with them. This is made more difficult by the multiple comparison problem due to the construction one interval for each parameter!
- Be careful with interpretation of coefficients and HPD intervals if different columns of the response matrix have different distributions!
- HPD intervals for the cutoffs in proportional odds regression may be poorly estimated for levels with few data.

Note

`boral` fits the model and returns the HPD intervals by default.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]
Maintainer: Francis K.C. Hui <fhui28@gmail.com>

Examples

```
## Not run:
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1 - model with two latent variables, site effects,
##  and no environmental covariates
spiderfit_nb <- boral(y, family = "negative.binomial",
    lv.control = list(num.lv = 2), row.eff = "fixed",
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## Example 2a - model with no latent variables, no site effects,
##      and environmental covariates
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## Example 2b - suppose now, for some reason, the 28 rows were
##  sampled such into four replications of seven sites
## Let us account for this as a fixed effect
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    row.eff = "fixed", row.ids = matrix(rep(1:7,each=4),ncol=1),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## Example 2c - suppose now, for some reason, the 28 rows reflected
##  a nested design with seven regions, each with four sub-regions
## We can account for this nesting as a random effect
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    row.eff = "random",
    row.ids = cbind(1:n, rep(1:7,each=4)),
```

```
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## Example 2d - model with environmental covariates and
##  two structured latent variables using fake distance matrix
fakedistmat <- as.matrix(dist(1:n))
spiderfit_lvstruc <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2, type = "exponential", distmat = fakedistmat),
     mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## Example 2e - Similar to 2d, but we will species-specific random intercepts
##   for the seven regions (with row effects in the model)
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$hpdintervals

## End(Not run)
```

`get.mcmcsamples` *Extract MCMC samples from models*

Description

[Stable]

Extract the MCMC samples from fitted models, taking into account the burnin period and thinning.

Usage

```
get.mcmcsamples(object)
```

Arguments

`object`	An object for class "boral".

Details

For the function to work, the JAGS model file (containing the MCMC samples from the call to JAGS) has to have been saved when fitting the model, that is, `save.model = TRUE`. The function will throw an error if it cannot find the the JAGs model file.

Value

A matrix containing the MCMC samples, with the number of rows equal to the number of MCMC samples after accounting the burnin period and thinning (i.e., number of rows = (n.iteration - n.burnin)/n.thin), and the number of columns equal to the number of parameters in the fitted model.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
library(corrplot) ## For plotting correlations
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
     mcmc.control = example_mcmc_control, model.name = testpath,
     save.model = TRUE)

mcmcsamps <- get.mcmcsamples(spiderfit_nb)

## End(Not run)
```

`get.measures` *Information Criteria for models*

Description

[Defunct]

Calculates some information criteria for a fitted model, which could be used for model selection. WARNING: As of version 1.6, this function is no longer maintained (and probably doesn't work properly, if at all)!

Usage

```
get.measures(y, X = NULL, family, trial.size = 1, row.eff = "none",
 row.ids = NULL, offset = NULL, num.lv, fit.mcmc)
```

Arguments

y	The response matrix that the model was fitted to.
X	The covariate matrix used in the model. Defaults to NULL, in which case it is assumed no model matrix was used.
family	Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link). Please see about.distributions for information on distributions available in boral overall.
trial.size	Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.
row.eff	Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and estimated standard deviation. Defaults to "none".
row.ids	A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see boral for details. Defaults to NULL, so that if row.eff = "none" then the argument is ignored, otherwise if row.eff = "fixed" or "random", then row.ids = matrix(1:nrow(y), ncol = 1) i.e., a single, row effect unique to each row.
offset	A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to NULL.
num.lv	The number of latent variables used in the model.
fit.mcmc	All MCMC samples for the fitted model. These can be extracted by fitting a model using boral with save.model = TRUE, and then applying get.mcmcsamples(fit).

Details

The following information criteria are currently calculated, when permitted: 1) Widely Applicable Information Criterion (WAIC, Watanabe, 2010) based on the conditional log-likelihood; 2) expected AIC (EAIC, Carlin and Louis, 2011); 3) expected BIC (EBIC, Carlin and Louis, 2011); 4)

AIC (using the marginal likelihood) evaluated at the posterior median; 5) BIC (using the marginal likelihood) evaluated at the posterior median.

1) WAIC has been argued to be more natural and extension of AIC to the Bayesian and hierarchical modeling context (Gelman et al., 2013), and is based on the conditional log-likelihood calculated at each of the MCMC samples.

2 & 3) EAIC and EBIC were suggested by (Carlin and Louis, 2011). Both criteria are of the form -2*mean(conditional log-likelihood) + penalty*(no. of parameters in the model), where the mean is averaged all the MCMC samples. EAIC applies a penalty of 2, while EBIC applies a penalty of $log(n)$.

4 & 5) AIC and BIC take the form -2*(marginal log-likelihood) + penalty*(no. of parameters in the model), where the log-likelihood is evaluated at the posterior median. If the parameter-wise posterior distributions are unimodal and approximately symmetric, these will produce similar results to an AIC and BIC where the log-likelihood is evaluated at the posterior mode. EAIC applies a penalty of 2, while EBIC applies a penalty of $log(n)$.

Intuitively, comparing models with and without latent variables (using information criteria such as those returned) amounts to testing whether the columns of the response matrix are correlated. With multivariate abundance data for example, where the response matrix comprises of n sites and p species, comparing models with and without latent variables tests whether there is any evidence of correlation between species.

Please note that criteria 4 and 5 are not calculated all the time. In models where traits are included in the model (such that the regression coefficients $\beta_{0j}, \boldsymbol{\beta}_j$ are random effects), or more than two columns are ordinal responses (such that the intercepts β_{0j} for these columns are random effects), then criteria 4 and 5 are will not calculated. This is because the calculation of the marginal log-likelihood in such cases currently fail to marginalize over such random effects; please see the details in `calc.logLik.lv0` and `calc.marglogLik`.

Value

A list with the following components:

`waic` WAIC based on the conditional log-likelihood.

`eaic` EAIC based on the mean of the conditional log-likelihood.

`ebic` EBIC based on the mean of the conditional log-likelihood.

`all.cond.logLik`
The conditional log-likelihood evaluated at all MCMC samples. This is done via repeated application of `calc.condlogLik`.

`cond.num.params`
Number of estimated parameters used in the fitted model, when all parameters are treated as "fixed" effects.

`do.marglik.ics` A boolean indicating whether marginal log-likelihood based information criteria are calculated.

If `do.marglik.ics = TRUE`, then we also have:

`median.logLik` The marginal log-likelihood evaluated at the posterior median.

`marg.num.params`
Number of estimated parameters used in the fitted model, when all parameters are treated as "fixed" effects.

aic.median	AIC (using the marginal log-likelihood) evaluated at the posterior median.
bic.median	BIC (using the marginal log-likelihood) evaluated at the posterior median.

Warning

As of version 1.6, this function is no longer maintained (and probably doesn't work properly, if at all)!

Using information criterion for variable selection should be done with extreme caution, for two reasons: 1) The implementation of these criteria are both *heuristic* and experimental. 2) Deciding what model to fit for ordination purposes should be driven by the science. For example, it may be the case that a criterion suggests a model with 3 or 4 latent variables. However, if we interested in visualizing the data for ordination purposes, then models with 1 or 2 latent variables are far more appropriate. As an another example, whether or not we include row effects when ordinating multivariate abundance data depends on if we are interested in differences between sites in terms of relative species abundance (`row.eff = FALSE`) or in terms of species composition (`row.eff = "fixed"`).

Also, the use of information criterion in the presence of variable selection using SSVS is questionable.

Note

When a model is fitted using `boral` with `calc.ics = TRUE`, then this function is applied and the information criteria are returned as part of the model output.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Carlin, B. P., and Louis, T. A. (2011). Bayesian methods for data analysis. CRC Press.
- Gelman et al. (2013). Understanding predictive information criteria for Bayesian models. Statistics and Computing, 1-20.
- Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. The Journal of Machine Learning Research, 11, 3571-3594.

See Also

`get.dic` for calculating the Deviance Information Criterion (DIC) based on the conditional log-likelihood; `get.more.measures` for even more information criteria.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
```

```
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

spiderfit_pois <- boral(y, family = "poisson",
    lv.control = list(num.lv = 2), row.eff = "random",
    mcmc.control = example_mcmc_control)

spiderfit_pois$ics ## Returns information criteria

spiderfit_nb <- boral(y, family = "negative.binomial",
    lv.control = list(num.lv = 2), row.eff = "random",
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb$ics ## Returns the information criteria

## End(Not run)
```

`get.more.measures` *Additional Information Criteria for models*

Description

[Defunct]

Calculates some information criteria beyond those from `get.measures` for a fitted model, although this set of criteria takes much longer to compute! WARNING: As of version 1.6, this function is no longer maintained (and probably doesn't work properly, if at all)!

Usage

```
get.more.measures(y, X = NULL, family, trial.size = 1,
 row.eff = "none", row.ids = NULL, offset = NULL,
 num.lv, fit.mcmc, verbose = TRUE)
```

Arguments

`y`	The response matrix that the model was fitted to.
`X`	The covariate matrix used in the model. Defaults to `NULL`, in which case it is assumed no model matrix was used.

`family` Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link).

Please see `about.distributions` for information on distributions available in boral overall.

`trial.size` Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

`row.eff` Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and estimated standard deviation. Defaults to "none".

`row.ids` A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see `boral` for details. Defaults to `NULL`, so that if `row.eff = "none"` then the argument is ignored, otherwise if `row.eff = "fixed"` or `"random"`, then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row.

`offset` A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to `NULL`.

`num.lv` The number of latent variables used in the fitted model.

`fit.mcmc` All MCMC samples for the fitted model. These can be extracted by fitting a model using `boral` with `save.model = TRUE`, and then applying `get.mcmcsamples(fit)`.

`verbose` If TRUE, a notice is printed every 100 samples indicating progress in calculation of the marginal log-likelihood. Defaults to `TRUE`.

Details

Currently, four information criteria are calculated using this function, when permitted: 1) AIC (using the marginal likelihood) evaluated at the posterior mode; 2) BIC (using the marginal likelihood) evaluated at the posterior mode; 3) Deviance information criterion (DIC) based on the marginal log-likelihood; 4) Widely Applicable Information Criterion (WAIC, Watanabe, 2010) based on the

marginal log-likelihood. When uninformative priors are used in fitting models, then the posterior mode should be approximately equal to the maximum likelihood estimates.

All four criteria require computing the marginal log-likelihood across all MCMC samples. This takes a very long time to run, since Monte Carlo integration needs to be performed for all MCMC samples. Consequently, this function is currently not implemented as an argument in main `boral` fitting function, unlike `get.measures` which is available via the `calc.ics = TRUE` argument.

Moreover, note these criteria are not calculated all the time. In models where traits are included in the model (such that the regression coefficients $\beta_{0j}, \boldsymbol{\beta}_j$ are random effects), or more than two columns are ordinal responses (such that the intercepts β_{0j} for these columns are random effects), then these extra information criteria are will not calculated, and the function returns nothing except a simple message. This is because the calculation of the marginal log-likelihood in such cases currently fail to marginalize over such random effects; please see the details in `calc.logLik.lv0` and `calc.marglogLik`.

The two main differences between the criteria and those returned from `get.measures` are:

- The AIC and BIC computed here are based on the log-likelihood evaluated at the posterior mode, whereas the AIC and BIC from `get.measures` are evaluated at the posterior median. The posterior mode and median will be quite close to one another if the component-wise posterior distributions are unimodal and symmetric. Furthermore, given uninformative priors are used, then both will be approximate maximum likelihood estimators.
- The DIC and WAIC computed here are based on the marginal log-likelihood, whereas the DIC and WAIC from `get.measures` are based on the conditional log-likelihood. Criteria based on the two types of log-likelihood are equally valid, and to a certain extent, which one to use depends on the question being answered i.e., whether to condition on the latent variables or treat them as "random effects" (see discussions in Spiegelhalter et al. 2002, and Vaida and Blanchard, 2005).

Value

If calculated, then a list with the following components:

`marg.aic`	AIC (using on the marginal log-likelihood) evaluated at posterior mode.
`marg.bic`	BIC (using on the marginal log-likelihood) evaluated at posterior mode.
`marg.dic`	DIC based on the marginal log-likelihood.
`marg.waic`	WAIC based on the marginal log-likelihood.
`all.marg.logLik`	The marginal log-likelihood evaluated at all MCMC samples. This is done via repeated application of `calc.marglogLik`.
`num.params`	Number of estimated parameters used in the fitted model.

Warning

As of version 1.6, this function is no longer maintained (and probably doesn't work)!

Using information criterion for variable selection should be done with extreme caution, for two reasons: 1) The implementation of these criteria are both *heuristic* and experimental. 2) Deciding what model to fit for ordination purposes should be driven by the science. For example, it may be the case that a criterion suggests a model with 3 or 4 latent variables. However, if we interested

in visualizing the data for ordination purposes, then models with 1 or 2 latent variables are far more appropriate. As an another example, whether or not we include row effects when ordinating multivariate abundance data depends on if we are interested in differences between sites in terms of relative species abundance (`row.eff = FALSE`) or in terms of species composition (`row.eff = "fixed"`).

Also, the use of information criterion in the presence of variable selection using SSVS is questionable.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Spiegelhalter et al. (2002). Bayesian measures of model complexity and fit. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64, 583-639.
- Vaida, F., and Blanchard, S. (2005). Conditional Akaike information for mixed-effects models. Biometrika, 92, 351-370.
- Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. The Journal of Machine Learning Research, 11, 3571-3594.

See Also

`get.measures` for several information criteria which take considerably less time to compute, and are automatically implemented in `boral` with `calc.ics = TRUE`.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
    n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
    row.eff = "fixed", save.model = TRUE, calc.ics = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)

## Extract MCMC samples
```

```
fit_mcmc <- get.mcmcsamples(spiderfit_nb)

## NOTE: The following takes a long time to run!
get.more.measures(y, family = "negative.binomial",
    num.lv = spiderfit_nb$num.lv, fit.mcmc = fit_mcmc,
    row.eff = "fixed", row.ids = spiderfit_nb$row.ids)


## Illustrating what happens in a case where these criteria will
##  not be calculated.
data(antTraits)
y <- antTraits$abun
X <- as.matrix(scale(antTraits$env))
## Include only traits 1, 2, and 5
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)

fit_traits <- boral(y, X = X, traits = traits, lv.control = list(num.lv = 2),
    which.traits = example_which_traits, family = "negative.binomial",
    save.model = TRUE, mcmc.control = example_mcmc_control,
    model.name = testpath)

## Extract MCMC samples
fit_mcmc <- get.mcmcsamples(fit_traits)

get.more.measures(y, X = X, family = "negative.binomial",
    num.lv = fit_traits$num.lv, fit.mcmc = fit_mcmc)

## End(Not run)
```

`get.residual.cor` *Extract residual correlations and precisions from models*

Description

[Stable]

Calculates the residual correlation and precision matrices from models that include latent variables.

Usage

```
get.residual.cor(object, est = "median", prob = 0.95)
```

Arguments

`object`	An object for class "boral".
`est`	A choice of either the posterior median (`est = "median"`) or posterior mean (`est = "mean"`), which are then treated as estimates and the fitted values are calculated from. Default is posterior median.

prob A numeric scalar in the interval (0,1) giving the target probability coverage of the intervals, by which to determine whether the correlations and precisions are "significant". Defaults to 0.95.

Details

In models with latent variables, the residual covariance matrix is calculated based on the matrix of latent variables regression coefficients formed by stacking the rows of $\boldsymbol{\theta}_j$. That is, if we denote $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1 \dots \boldsymbol{\theta}_p)'$, then the residual covariance and hence residual correlation matrix is calculated based on $\boldsymbol{\Theta}\boldsymbol{\Theta}'$.

For multivariate abundance data, the inclusion of latent variables provides a parsimonious method of accounting for correlation between species. Specifically, the linear predictor,

$$\beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{u}_i^\top \boldsymbol{\theta}_j$$

is normally distributed with a residual covariance matrix given by $\boldsymbol{\Theta}\boldsymbol{\Theta}'$. A strong residual covariance/correlation matrix between two species can then be interpreted as evidence of species interaction (e.g., facilitation or competition), missing covariates, as well as any additional species correlation not accounted for by shared environmental responses (see also Pollock et al., 2014, for residual correlation matrices in the context of Joint Species Distribution Models). If random effects $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ are also included in the model, then the latent variables induce a residual covariance/correlation between responses than is conditional on this.

The residual precision matrix (also known as partial correlation matrix, Ovaskainen et al., 2016) is defined as the inverse of the residual correlation matrix. The precision matrix is often used to identify direct or causal relationships between two species e.g., two species can have a zero precision but still be correlated, which can be interpreted as saying that two species do not directly interact, but they are still correlated through other species. In other words, they are conditionally independent given the other species. It is important that the precision matrix does not exhibit the exact same properties of the correlation e.g., the diagonal elements are not equal to 1. Nevertheless, relatively larger values of precision imply a stronger direct relationships between two species.

In addition to the residual correlation and precision matrices, the median or mean point estimator of trace of the residual covariance matrix is returned, $\sum_{j=1}^{p} [\boldsymbol{\Theta}\boldsymbol{\Theta}']_{jj}$. Often used in other areas of multivariate statistics, the trace may be interpreted as the amount of covariation explained by the latent variables. One situation where the trace may be useful is when comparing a pure LVM versus a model with latent variables and some predictors (correlated response models) – the proportional difference in trace between these two models may be interpreted as the proportion of covariation between species explained by the predictors. Of course, the trace itself is random due to the MCMC sampling, and so it is not always guranteed to produce sensible answers!

Value

A list with the following components:

cor, cor.lower, cor.upper
A set of $p \times p$ correlation matrices, containing either the posterior median or mean estimate plus lower and upper limits of the corresponding (100×prob) % HPD interval.

`sig.cor` A $p \times p$ correlation matrix containing only the "significant" correlations whose (100×`prob`) % HPD interval does not contain zero. All non-significant correlations are set to zero.

`cov` A $p \times p$ covariance matrix.

`prec`, `prec.lower`, `prec.upper`
A set of $p \times p$ precision matrices, containing either the posterior median or mean estimate plus lower and upper limits of the corresponding (100×`prob`) % HPD interval.

`sig.prec` A $p \times p$ residual precision matrix containing only the "significant" precisions whose (100×`prob`) % HPD interval does not contain zero. All non-significant precision are set to zero.

`trace` The median/mean point estimator of the trace (sum of the diagonal elements) of the residual covariance matrix.

Note

Residual correlation and precision matrices are reliably modeled only with two or more latent variables i.e., `num.lv` > 1 when fitting the model using `boral`.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Ovaskainen et al. (2016). Using latent variable models to identify large networks of species-to-species associations at different spatial scales. Methods in Ecology and Evolution, 7, 549-555.
- Pollock et al. (2014). Understanding co-occurrence by modelling species simultaneously with a Joint Species Distribution Model (JSDM). Methods in Ecology and Evolution, 5, 397-406.

See Also

`get.enviro.cor`, which calculates the correlation matrix due to similarities in the response to the explanatory variables (i.e., similarities due to a shared environmental response).

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
```

```
library(corrplot) ## For plotting correlations
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

spiderfit_nb <- boral(y, X = spider$x, family = "negative.binomial",
    lv.control = list(num.lv = 2), save.model = TRUE,
    mcmc.control = example_mcmc_control, model.name = testpath)

res.cors <- get.residual.cor(spiderfit_nb)

corrplot(res.cors$sig.cor, title = "Residual correlations",
    type = "lower", diag = FALSE, mar = c(3,0.5,2,1), tl.srt = 45)

## End(Not run)
```

`lvsplot` *Plot the latent variables from a fitted model*

Description

[Stable]

Construct a 1-D index plot or 2-D scatterplot of the latent variables, and their corresponding coefficients i.e., a biplot, from a fitted model.

Usage

```
lvsplot(object, jitter = FALSE, biplot = TRUE, ind.spp = NULL, alpha = 0.5,
 main = NULL, est = "median", which.lvs = c(1,2), return.vals = FALSE, ...)
```

Arguments

`object`	An object for class "boral".
`jitter`	If `jitter = TRUE`, then some jittering is applied so that points on the plots do not overlap exactly (which can often occur with discrete data, small sample sizes, and if some sites are identical in terms species co-occurence). Please see `jitter` for its implementation. Defaults to `FALSE`.
`biplot`	If `biplot = TRUE`, then a biplot is construct such that both the latent variables *and* their corresponding coefficients are plotted. Otherwise, only the latent variable scores are plotted. Defaults to `TRUE`.
`ind.spp`	Controls the number of latent variable coefficients to plot if `biplot = TRUE`. If `ind.spp` is an integer, then only the first `ind.spp` "most important" latent variable coefficients are included in the biplot, where "most important" means the latent variable coefficients with the largests L2-norms. Defaults to `NULL`, in which case all latent variable coefficients are included in the biplot.

alpha	A numeric scalar between 0 and 1 that is used to control the relative scaling of the latent variables and their coefficients, when constructing a biplot. Defaults to 0.5, and we typically recommend between 0.45 to 0.55 so that the latent variables and their coefficients are on roughly the same scale.
main	Title for resulting ordination plot. Defaults to NULL, in which case a "standard" title is used.
est	A choice of either the posterior median (est = "median") or posterior mean (est = "mean"), which are then treated as estimates and the ordinations based off. Default is posterior median.
which.lvs	A vector of length two, indicating which latent variables (ordination axes) to plot which object is an object with two or more latent variables. The argument is ignored is object only contains one latent variables. Defaults to which.lvs = c(1,2).
return.vals	If TRUE, then the *scaled* latent variables scores and corresponding scaled coefficients are returned (based on the value of alpha used). This is useful if the user wants to construct their own custom model-based ordinations. Defaults to FALSE.
...	Additional graphical options to be included in. These include values for cex, cex.lab, cex.axis, cex.main, lwd, and so on.

Details

This function allows an ordination plot to be constructed, based on either the posterior medians and posterior means of the latent variables respectively depending on the choice of est. The latent variables are labeled using the row index of the response matrix. If the fitted model contains more than two latent variables, then one can specify which latent variables i.e., ordination axes, to plot based on the which.lvs argument. This can prove useful (to check) if certain sites are outliers on one particular ordination axes.

If the fitted model did not contain any covariates, the ordination plot can be interpreted in the exactly same manner as unconstrained ordination plots constructed from methods such as Nonmetric Multidimensional Scaling (NMDS, Kruskal, 1964) and Correspondence Analysis (CA, Hill, 1974). With multivariate abundance data for instance, where the response matrix comprises of n sites and p species, the ordination plots can be studied to look for possible clustering of sites, location and/or dispersion effects, an arch pattern indicative of some sort species succession over an environmental gradient, and so on.

If the fitted model did include covariates, then a “residual ordination" plot is produced, which can be interpreted can offering a graphical representation of the (main patterns of) residual covarations, i.e. covariations after accounting for the covariates. With multivariate abundance data for instance, these residual ordination plots represent could represent residual species co-occurrence due to phylogency, species competition and facilitation, missing covariates, and so on (Warton et al., 2015)

If biplot = TRUE, then a biplot is constructed so that both the latent variables and their corresponding coefficients are included in their plot (Gabriel, 1971). The latent variable coefficients are shown in red, and are indexed by the column names of the response matrix. The number of latent variable coefficients to plot is controlled by ind.spp. In ecology for example, often we are only be interested in the "indicator" species, e.g. the species with most represent a particular set of sites

or species with the strongest covariation (see Chapter 9, Legendre and Legendre, 2012, for additional discussion). In such case, we can then biplot only the `ind.spp` "most important" species, as indicated by the the L2-norm of their latent variable coefficients.

As with correspondence analysis, the relative scaling of the latent variables and the coefficients in a biplot is essentially arbitrary, and could be adjusted to focus on the sites, species, or put even weight on both (see Section 9.4, Legendre and Legendre, 2012). In `lvsplot`, this relative scaling is controlled by the `alpha` argument, which basically works by taking the latent variables to a power `alpha` and the latent variable coefficients to a power `1-alpha`.

For latent variable models, we are generally interested in "symmetric plots" that place the latent variables and their coefficients on the same scale. In principle, this is achieved by setting `alpha = 0.5`, the default value, although sometimes this needs to be tweaked slighlty to a value between 0.45 and 0.55 (see also the `corresp` function in the `MASS` package that also produces symmetric plots, as well as Section 5.4, Borcard et al., 2011 for more details on scaling).

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Borcard et al. (2011). Numerical Ecology with R. Springer.
- Gabriel, K. R. (1971). The biplot graphic display of matrices with application to principal component analysis. Biometrika, 58, 453-467.
- Hill, M. O. (1974). Correspondence analysis: a neglected multivariate method. Applied statistics, 23, 340-354.
- Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. Psychometrika, 29, 115-129.
- Legendre, P. and Legendre, L. (2012). Numerical ecology, Volume 20. Elsevier.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, to appear

Examples

```
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)
```

```
spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
    row.eff = "fixed", mcmc.control = example_mcmc_control, model.name = testpath)

lvsplot(spiderfit_nb)
```

`make.jagsboralmodel` *Write a text file containing a model for use into JAGS*

Description

[Stable]

This function is designed to write models with one or more latent variables.

Usage

```
make.jagsboralmodel(family, num.X = 0, X.ind =  NULL, num.traits = 0,
     which.traits = NULL, lv.control = list(num.lv = 2, type = "independent"),
     row.eff = "none", row.ids = NULL, ranef.ids = NULL,
     offset = NULL, trial.size = 1, n, p, model.name = NULL,
     prior.control = list(type = c("normal","normal","normal","uniform"),
     hypparams = c(10, 10, 10, 30), ssvs.index = -1, ssvs.g = 1e-6,
 ssvs.traitsindex = -1),
     num.lv = NULL)
```

Arguments

`family`	Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link). Please see `about.distributions` for information on distributions available in boral overall.
`num.X`	Number of columns in the covariate matrix. Defaults to 0, in which case it is assumed that no covariates are included in the model. Recall that no intercept is included in the covariate matrix.
`X.ind`	An matrix of 1s and 0s, indicating whether a particular covariate should be included (1) or excluded (0) in the mean structure of a particular response. The matrix should the number of rows equal to the number of columns in the response matrix, and the number of columns equal to the number of columns in the covariate matrix. Defaults to `NULL`, in which case it is assumed that all covariates are included in the mean structure of all responses i.e., all 1s.

num.traits — Number of columns in the trait matrix. Defaults to 0, in which case it is assumed no traits are included in model. Recall that no intercept should is included in the trait matrix.

which.traits — A list of length equal to (number of columns in the covariate matrix + 1), informing which columns of the trait matrix the response-specific intercepts and each of the response-specific regression coefficients should be regressed against. The first element in the list applies to the response-specific intercept, while the remaining elements apply to the regression coefficients. Each element of `which.traits` is a vector indicating which traits are to be used.

For example, if `which.traits[[2]] = c(2,3)`, then the regression coefficients corresponding to the first column in the covariate matrix are regressed against the second and third columns of the trait matrix. If `which.traits[[2]][1] = 0`, then the regression coefficients for each column are treated as independent. Please see `about.traits` for more details.

Defaults to `NULL`, and used in conjunction with `traits` and `prior.control$ssvs.traitsindex`.

lv.control — A list (currently) with the following arguments:

- *num.lv:* which specifies the number of true latent variables to generate. Defaults to 0.
- *type:* which specifies the type the correlation structure of the latent variables (across sites). Defaults to independence correlation structure.

Please see `about.lvs` for more information.

row.eff — Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and unknown standard deviation. Defaults to "none".

row.ids — A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see `boral` for details. Defaults to `NULL`, so that if `row.eff = "none"` then the argument is ignored, otherwise if `row.eff = "fixed"` or `"random"`, then `row.ids = matrix(1:nrow(y), ncol = 1)` i.e., a single, row effect unique to each row.

ranef.ids — A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j; please see `about.ranefs` for details. Defaults to `NULL`, in which case it is assumed no random intercepts are to be included in the model. If supplied, then response-specific random intercepts are assumed to come from a normal distribution with mean zero and unknown (response-specific) standard deviation.

offset — A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to `NULL`.

`trial.size` Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

`n` The number of rows in the response matrix.

`p` The number of columns in the response matrix.

`model.name` Name of the text file that the JAGS script is written to. Defaults to `NULL`, in which case the default of "jagsboralmodel.txt" is used.

`prior.control` A list of parameters for controlling the prior distributions. These include:

- *type:* Vector of four strings indicating the type of prior distributions to use. In order, these are: 1) priors for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) priors for the latent variable coefficients and covariance parameters. This is ignored if `lv.control$num.lv = 0`; 3) priors for all response-specific coefficients relating to the covariate matrix (ignored if `X = NULL`). When traits are included in the model, this is also the prior for the trait regression coefficients (please see `about.traits` for more information); 4) priors for any dispersion parameters and variance (standard deviation, to be precise) parameters in the model.
  For elements 1-3, the prior distributions currently available include: I) "normal", which is a normal prior with the variance controlled by elements 1-3 in `hypparams`; II) "cauchy", which is a Cauchy prior with variance controlled by elements 1-3 in `hypparams`. Gelman, et al. (2008) considers using Cauchy priors with variance 2.5^2 as weakly informative priors for coefficients in logistic and potentially other generalized linear models; III) "uniform", which is a symmetric uniform prior with minimum and maximum values controlled by element 1-3 in `hypparams`.
  For element 4, the prior distributions currently available include: I) "uniform", which is uniform prior with minimum zero and maximum controlled by element 4 in `hypparmas`; II) "halfnormal", which is half-normal prior with variance controlled by `hypparams`; III) "halfcauchy", which is a half-Cauchy prior with variance controlled by element 4 in `hypparams`.
  Defaults to the vector `c("normal","normal","normal","uniform")`.
- *hypparams:* Vector of four hyperparameters used in the set up of prior distributions. In order, these are: 1) affects the prior distribution for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) affects the prior distribution for all latent variable coefficients and correlation parameters. This is ignored if `lv.control$num.lv = 0`; 3) affects the prior distribution for response-specific coefficients relating to the covariate matrix (ignored if `X = NULL`). When traits are included in the model, it also affects the prior distribution for the trait regression coefficients; 4) affects the prior distribution for any dispersion parameters, as well as the prior distributions for the standard deviation of the random effects normal distribution if `row.eff = "random"`, the standard deviation of the response-specific random intercepts for these columns if more than two of the columns are ordinal, and the standard deviation of the random effects normal distribution for trait regression coefficients when traits are included in the model.

Defaults to the vector `c(10, 10, 10, 30)`. The use of normal distributions with mean zero and variance 10 as priors is seen as one type of (very) weakly informative prior, according to Prior choice recommendations.

- *ssvs.index:* Indices to be used for stochastic search variable selection (SSVS, George and McCulloch, 1993). Either a single element or a vector with length equal to the number of columns in covariate matrix. Each element can take values of -1 (no SSVS is performed on this covariate), 0 (SSVS is performed on individual coefficients for this covariate), or any integer greater than 0 (SSVS is performed on collectively all coefficients on this covariate/s.)
  Please see `about.ssvs` for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on the covariates.
- *ssvs.g:* Multiplicative, shrinkage factor for SSVS, which controls the strength of the "spike" in the SSVS mixture prior. In summary, if the coefficient is included in the model, the "slab" prior is a normal distribution with mean zero and variance given by element 3 in `hypparams`, while if the coefficient is not included in the model, the "spike" prior is normal distribution with mean zero and variance given by element 3 in `hypparams` multiplied by `ssvs.g`. Please see `about.ssvs` for more information regarding the implementation of SSVS. Defaults to 1e-6.
- *ssvs.traitsindex:* Used in conjunction with `traits` and `which.traits`, this is a list of indices to be used for performing SSVS on the trait coefficients. Should be a list with the same length as `which.traits`, and with each element a vector of indices with the same length as the corresponding element in `which.traits`. Each index either can take values of -1 (no SSVS on this trait coefficient) or 0 (no SSVS on this trait coefficient).
  Please see `about.ssvs` for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on any of the trait coefficients, if they are included in the model.

`num.lv` Old argument superceded by `lv.control`. Defaults to `NULL` and ignored.

Details

This function is automatically executed inside `boral`, and therefore does not need to be run separately before fitting the model. It can however be run independently if one is: 1) interested in what the actual JAGS file for a particular model looks like, 2) wanting to modify a basic JAGS model file to construct more complex model e.g., include environmental variables.

Please note that `boral` currently does not allow the user to manually enter a script to be run.

When running the main function `boral`, setting `save.model = TRUE` which automatically save the JAGS model file as a text file (with name based on the `model.name`) in the current working directory.

Value

A text file is created, containing the model to be called by the boral function for entering into JAGS. This file is automatically deleted once boral has finished running `save.model = TRUE`.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Gelman et al. (2008). A weakly informative default prior distribution for logistic and other regression models. The Annals of Applied Statistics, 2, 1360-1383.

See Also

make.jagsboralnullmodel for writing JAGS scripts for models with no latent variables i.e., so-called "null models".

Examples

```
library(mvtnorm)
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1 - Create a JAGS model file, where distributions alternative
## between Poisson and negative binomial distributions
##   across the rows of y.
make.jagsboralmodel(family = rep(c("poisson","negative.binomial"),length=p),
    row.eff = "fixed", num.X = 0, n = n, p = p, model.name = testpath)


## Example 2 - Create a JAGS model file, where distributions are all
## negative binomial distributions and covariates will be included.
make.jagsboralmodel(family = "negative.binomial", num.X = ncol(spider$x),
    n = n, p = p, model.name = testpath)


## Example 3 - Simulate some ordinal data and create a JAGS model file
## 30 rows (sites) with two latent variables
true.lv <- rbind(rmvnorm(15,mean=c(-2,-2)),rmvnorm(15,mean=c(2,2)))
## 10 columns (species)
true.lv.coefs <- rmvnorm(10,mean = rep(0,3));
true.lv.coefs[nrow(true.lv.coefs),1] <- -sum(true.lv.coefs[-nrow(true.lv.coefs),1])
## Impose a sum-to-zero constraint on the column effects
true.ordinal.cutoffs <- seq(-2,10,length=10-1)

simy <- create.life(true.lv = true.lv, lv.coefs = true.lv.coefs,
    family = "ordinal", cutoffs = true.ordinal.cutoffs)
```

```
make.jagsboralmodel(family = "ordinal", num.X = 0,
    row.eff = FALSE, n=30, p=10, model.name = testpath)


## Have a look at the JAGS model file for a model involving traits,
## based on the ants data from mvabund.
library(mvabund)
data(antTraits)

y <- antTraits$abun
X <- as.matrix(antTraits$env)
## Include only traits 1, 2, and 5, plus an intercept
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
## Please see help file for boral regarding the use of which.traits
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
    example_which_traits[[i]] <- 1:ncol(traits)

## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
    n.thin = 1)

fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", lv.control = list(num.lv = 2),
    model.name = testpath, mcmc.control = example_mcmc_control,
    do.fit = FALSE)

## End(Not run)
```

`make.jagsboralnullmodel` *Write a text file containing a model for use into JAGS*

Description

[Stable]

This function is designed to write models with no latent variables i.e., so-called "null" models.

Usage

```
make.jagsboralnullmodel(family, num.X = 0, X.ind = NULL, num.traits = 0,
     which.traits = NULL, row.eff = "none", row.ids = NULL, ranef.ids = NULL,
     offset = NULL, trial.size = 1, n, p, model.name = NULL,
     prior.control = list(type = c("normal","normal","normal","uniform"),
 hypparams = c(10, 10, 10, 30), ssvs.index = -1, ssvs.g = 1e-6,
 ssvs.traitsindex = -1))
```

Arguments

family	Either a single element, or a vector of length equal to the number of columns in the response matrix. The former assumes all columns of the response matrix come from this distribution. The latter option allows for different distributions for each column of the response matrix. Elements can be one of "binomial" (with probit link), "poisson" (with log link), "negative.binomial" (with log link), "normal" (with identity link), "lnormal" for lognormal (with log link), "tweedie" (with log link), "exponential" (with log link), "gamma" (with log link), "beta" (with logit link), "ordinal" (cumulative probit regression), "ztpoisson" (zero truncated Poisson with log link), "ztnegative.binomial" (zero truncated negative binomial with log link). Please see about.distributions for information on distributions available in boral overall.
num.X	Number of columns in the covariate matrix. Defaults to 0, in which case it is assumed that no covariates are included in the model. Recall that no intercept is included in the covariate matrix.
X.ind	An matrix of 1s and 0s, indicating whether a particular covariate should be included (1) or excluded (0) in the mean structure of a particular response. The matrix should the number of rows equal to the number of columns in the response matrix, and the number of columns equal to the number of columns in the covariate matrix. Defaults to NULL, in which case it is assumed that all covariates are included in the mean structure of all responses i.e., all 1s.
num.traits	Number of columns in the trait matrix. Defaults to 0, in which case it is assumed no traits are included in model. Recall that no intercept is included in the trait matrix.
which.traits	A list of length equal to (number of columns in the covariate matrix + 1), informing which columns of the trait matrix the response-specific intercepts and each of the response-specific regression coefficients should be regressed against. The first element in the list applies to the response-specific intercept, while the remaining elements apply to the regression coefficients. Each element of which.traits is a vector indicating which traits are to be used. For example, if which.traits[[2]] = c(2,3), then the regression coefficients corresponding to the first column in the covariate matrix are regressed against the second and third columns of the trait matrix. If which.traits[[2]][1] = 0, then the regression coefficients for each column are treated as independent. Please see about.traits for more details. Defaults to NULL, and used in conjunction with traits and prior.control$ssvs.traitsindex.
row.eff	Single element indicating whether row effects are included as fixed effects ("fixed"), random effects ("random") or not included ("none") in the fitted model. If fixed effects, then for parameter identifiability the first row effect is set to zero, which analogous to acting as a reference level when dummy variables are used. If random effects, they are drawn from a normal distribution with mean zero and unknown standard deviation. Defaults to "none".
row.ids	A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of row effects to be

included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j; please see boral for more details. for details. Defaults to NULL, so that if row.eff = "none" then the argument is ignored, otherwise if
row.eff = "fixed" or "random",
then row.ids = matrix(1:nrow(y), ncol = 1) i.e., a single, row effect unique to each row.

ranef.ids — A matrix with the number of rows equal to the number of rows in the response matrix, and the number of columns equal to the number of random intercepts to be included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j; please see about.ranefs for details. Defaults to NULL, in which case it is assumed no random intercepts are to be included in the model. If supplied, then response-specific random intercepts are assumed to come from a normal distribution with mean zero and unknown (response-specific) standard deviation.

offset — A matrix with the same dimensions as the response matrix, specifying an a-priori known component to be included in the linear predictor during fitting. Defaults to NULL.

trial.size — Either equal to a single element, or a vector of length equal to the number of columns in y. If a single element, then all columns assumed to be binomially distributed will have trial size set to this. If a vector, different trial sizes are allowed in each column of y. The argument is ignored for all columns not assumed to be binomially distributed. Defaults to 1, i.e. Bernoulli distribution.

n — The number of rows in the response matrix.

p — The number of columns in the response matrix.

model.name — Name of the text file that the JAGS model is written to. Defaults to NULL, in which case the default of "jagsboralmodel.txt" is used.

prior.control — A list of parameters for controlling the prior distributions. These include:

- *type:* Vector of four strings indicating the type of prior distributions to use. In order, these are: 1) priors for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) priors for the latent variable coefficients and correlation parameters. This is ignored for this function; 3) priors for all response-specific coefficients relating to the covariate matrix (ignored if X = NULL). When traits are included in the model, this is also the prior for the trait regression coefficients (please see about.traits for more information); 4) priors for any dispersion parameters and variance (standard deviation, to be precise) parameters in the model.
  For elements 1-3, the prior distributions currently available include: I) "normal", which is a normal prior with the variance controlled by elements 1-3 in hypparams; II) "cauchy", which is a Cauchy prior with variance controlled by elements 1-3 in hypparams. Gelman, et al. (2008) considers using Cauchy priors with variance 2.5^2 as weakly informative priors for coefficients in logistic and potentially other generalized linear models; III) "uniform", which is a symmetric uniform prior with minimum and maximum values controlled by element 1-3 in hypparams.
  For element 4, the prior distributions currently available include: I) "uniform", which is uniform prior with minimum zero and maximum controlled

by element 4 in hypparmas; II) “halfnormal", which is half-normal prior with variance controlled by hypparams; III) “halfcauchy", which is a half-Cauchy prior with variance controlled by element 4 in hypparams.
Defaults to the vector c("normal","normal","normal","uniform").

- *hypparams* Vector of four hyperparameters used in the set up of prior distributions. In order, these are: 1) affects the prior distribution for all response-specific intercepts, row effects, and cutoff points for ordinal data; 2) affects the prior distribution for all latent variable coefficients and correlation parameters. This is ignored for this function; 3) affects the prior distribution for response-specific coefficients relating to the covariate matrix (ignored if X = NULL). When traits are included in the model, it also affects the prior distribution for the trait regression coefficients; 4) affects the prior distribution for any dispersion parameters, as well as the prior distributions for the standard deviation of the random effects normal distribution if row.eff = "random", the standard deviation of the response-specific random intercepts for these columns if more than two of the columns are ordinal, and the standard deviation of the random effects normal distribution for trait regression coefficients when traits are included in the model.
  Defaults to the vector c(10, 10, 10, 30). The use of normal distributions with mean zero and variance 10 as priors is seen as one type of (very) weakly informative prior, according to Prior choice recommendations.
- *ssvs.index:* Indices to be used for stochastic search variable selection (SSVS, George and McCulloch, 1993). Either a single element or a vector with length equal to the number of columns in the covariate matrix. Each element can take values of -1 (no SSVS is performed on this covariate), 0 (SSVS is performed on individual coefficients for this covariate), or any integer greater than 0 (SSVS is performed on collectively all coefficients on this covariate/s.)
  Please see about.ssvs for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on the covariates.
- *ssvs.g:* Multiplicative, shrinkage factor for SSVS, which controls the strength of the "spike" in the SSVS mixture prior. In summary, if the coefficient is included in the model, the "slab" prior is a normal distribution with mean zero and variance given by element 3 in hypparams, while if the coefficient is not included in the model, the "spike" prior is normal distribution with mean zero and variance given by element 3 in hypparams multiplied by ssvs.g. Please see about.ssvs for more information regarding the implementation of SSVS. Defaults to 1e-6.
- *ssvs.traitsindex:* Used in conjunction with traits and which.traits, this is a list of indices to be used for performing SSVS on the trait coefficients. Should be a list with the same length as which.traits, and with each element a vector of indices with the same length as the corresponding element in which.traits. Each index either can take values of -1 (no SSVS on this trait coefficient) or 0 (no SSVS on this trait coefficient).
  Please see about.ssvs for more information regarding the implementation of SSVS. Defaults to -1, in which case SSVS is not performed on any of the trait coefficients, if they are included in the model.

Details

This function is automatically executed inside boral, and therefore does not need to be run separately before fitting the model. It can however be run independently if one is: 1) interested in what the actual JAGS file for a particular model looks like, 2) wanting to modify a basic JAGS model file to construct more complex model e.g., include environmental variables.

Please note that boral currently does not allow the user to manually enter a script to be run.

When running the main function boral, setting save.model = TRUE which automatically save the JAGS model file as a text file (with name based on the model.name) in the current working directory.

Value

A text file is created, containing the JAGS model to be called by the boral function for entering into jags. This file is automatically deleted once boral has finished running unless save.model = TRUE.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Gelman et al. (2008). A weakly informative default prior distribution for logistic and other regression models. The Annals of Applied Statistics, 2, 1360-1383.

See Also

make.jagsboralmodel for writing JAGS scripts for models with one or more latent variables.

Examples

```
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
n <- nrow(y)
p <- ncol(y)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Create a "null" model JAGS script, where distributions alternative
## between Poisson and negative distributions
##   across the rows of y.
make.jagsboralnullmodel(family = rep(c("poisson","negative.binomial"),length=p),
    num.X = ncol(spider$x), row.eff = "fixed", n = n, p = p,
    model.name = testpath)


## Create a "null" model JAGS script, where distributions are all negative
##  binomial distributions and covariates will be included!
make.jagsboralnullmodel(family = "negative.binomial",
```

```
    num.X = ncol(spider$x), n = n, p = p,
    model.name = testpath)


## Have a look at the JAGS model file for a model involving traits,
## based on the ants data from mvabund.
library(mvabund)
data(antTraits)

y <- antTraits$abun
X <- as.matrix(antTraits$env)
## Include only traits 1, 2, and 5, plus an intercept
traits <- as.matrix(antTraits$traits[,c(1,2,5)])
## Please see help file for boral regarding the use of which.traits
example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)

## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

fit_traits <- boral(y, X = X, traits = traits, which.traits = example_which_traits,
    family = "negative.binomial", model.name = testpath,
     mcmc.control = example_mcmc_control, do.fit = FALSE)

## End(Not run)
```

`plot.boral` *Plots of a fitted boral object*

Description

[Stable]

Produces a set of four plots relating to the fitted boral object, which can be used for (some basic0 residual analysis. If some of the columns are ordinal, then a single confusion matrix is also produced.

Usage

```
## S3 method for class 'boral'
plot(x, est = "median", include.ranef = TRUE, jitter = FALSE, ...)
```

Arguments

x	An object of class "boral".

est	A choice of either the posterior median (est = "median") or posterior mean (est = "mean") of the parameters, which are then treated as parameter estimates and the fitted values/residuals used in the plots are calculated from. Default is posterior median.
jitter	If jitter = TRUE, then some jittering is applied so that points on the plots do not overlap exactly (which can often occur with discrete data). Please see jitter for its implementation.
include.ranef	If response-specific random intercepts were included as part of the fitted model, then this determines whether the predicted random effects will be used in the calculated of the fitted values and Dunn-Smyth residuals. When set to TRUE, which is the default, then they are included (using either the posterior mean and posterior median predictor). When set to FALSE, they are not included. The former leads to what are sometimes called conditional fitted values/residuals, while the latter are sometimes called marginal fitted values/residuals.
...	Additional graphical options to be included in. These include values for cex, cex.lab, cex.axis, cex.main, lwd, and so on.

Details

A set of four plots are provided:

1. Plot of Dunn-Smyth residuals against the linear predictors. For ordinal responses, things are more ambiguous due to the lack of single definition for "linear predictor". Therefore, instead of linear predictors the Dunn-Smyth residuals are plotted against the fitted values (defined as the level with the highest fitted probability). It is fully acknowledged that this makes things hard to interpret if only some of your columns are ordinal.
2. Plot of Dunn-Smyth residuals against the row index/names of the response matrix.
3. Plot of Dunn-Smyth residuals against the column index/names of the response matrix.
4. A normal quantile-quantile plot of the Dunn-Smyth residuals.

For ordinal responses, a single confusion matrix between the predicted levels (as based on the class with the highest probability) and true levels is aso returned. The table pools the results over all columns assumed to be ordinal.

Note

Due the inherent stochasticity, Dunn-Smyth residuals and consequently the plots will be slightly different time this function is run. Note also the fitted values and residuals are calculated from point estimates of the parameters, as opposed to a fully Bayesian approach (please see details in fitted.boral and ds.residuals). Consequently, it is recommended that this function is run several times to ensure that any trends observed in the plots are consistent throughout the runs.

As mentioned above, for ordinal responses things are much more challenging as there is no single definition for "linear predictor". Instead of linear predictors then, for the first plot the Dunn-Smyth residuals are plotted against the fitted values, defined as the level with the highest fitted probability. It is fully acknowledged that this makes things VERY hard to interpret if only some of your columns are ordinal though. Suggestions to improve this are welcome!!!

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

`fitted.boral` to obtain the fitted values, `ds.residuals` to obtain Dunn-Smyth residuals and details as to what they are.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
    n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun

spiderfit_pois <- boral(y, family = "poisson", lv.control = list(num.lv = 2),
    row.eff = "fixed", mcmc.control = example_mcmc_control
    model.name = testpath)

par(mfrow = c(2,2))
plot(spiderfit_pois)
## A distinct fan pattern is observed in the plot of residuals
## versus linear predictors plot.


spiderfit_nb <- boral(y, family = "negative.binomial", lv.control = list(num.lv = 2),
    row.eff = "fixed", mcmc.control = example_mcmc_control,
    model.name = testpath)

par(mfrow = c(2,2))
plot(spiderfit_nb)
## The fan shape is not as clear now,
## and the normal quantile plot also suggests a better fit to the data

## End(Not run)
```

`predict.boral` *Predict using a model*

Description

[Stable]

Construct predictions and associated intervals (lower and upper limits) from a fitted boral object. Predictions can be made either conditionally on the predicted latent variables and any random row effects/response-specific random intercepts included in the model, or marginally (averaged) on the latent variables and these other effects (note integration is done on the linear predictor scale).

Usage

```
## S3 method for class 'boral'
predict(object, newX = NULL, newrow.ids = NULL, newranef.ids = NULL,
     distmat =  NULL, predict.type = "conditional", scale = "link",
     est = "median", prob = 0.95, lv.mc = 1000, return.alllinpred = FALSE,
     ...)
```

Arguments

`object`	An object of class "boral".
`newX`	An optional model matrix of covariates for extrapolation to the same sites (under different environmental conditions) or extrapolation to new sites. No intercept column should be included in `newX`. Defaults to `NULL`, in which case the model matrix of covariates is taken from the fitted boral object if found.
`newrow.ids`	An optional matrix with the number of columns equal to the number of row effects included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random effect j. Defaults to `NULL`, in which case row IDs are taken from the fitted boral object itself, if appropriate, i.e., from `object$row.ids`.
`newranef.ids`	An optional matrix with the number of columns equal to the number of response-specific random intercepts included in the model. Element (i, j) indicates the cluster ID of row i in the response matrix for random intercept j. Defaults to `NULL`, in which case random intercept IDs are taken from the fitted boral object itself, if appropriate, i.e., from `object$ranef.ids`.
`distmat`	A distance matrix required to calculate correlations across sites when a non-independence correlation structure on the latent variables is imposed.
`predict.type`	The type of prediction to be made. Either takes value `"conditional"` in which case the prediction is made conditionally on the predicted latent variables and any random row effects in the model, or `"marginal"` in which case the prediction marginalizes (averages) over the latent variables and random row effects in the model. Defaults to `"conditional"`.
`scale`	The type of prediction required. The default is on the scale of the linear predictors; the alternative `scale = "response"` is on the scale of the response variable. For example, if the binomial family is used, then the default predictions (and associated uncertainty intervals) provide probabilities on linear predictor scale, while `scale = "response"` gives the predicted probabilities between 0 and 1.

Note things are slightly more complicated for zero truncated distributions because the log-link connects the mean of the *untruncated* distribution to the linear predictor. Therefore if `scale = "link"`, then the linear predictor is returned. But if `scale = "response"`, then actual predicted mean is returned.

`est` A choice of either whether to print the posterior median (`est = "median"`) or posterior mean (`est = "mean"`) of the parameters.

`prob` A numeric scalar in the interval (0,1) giving the target probability coverage of the intervals. Defaults to 0.95.

`lv.mc` If the predictions are made marginalizing over the latent variables, then number of Monte-Carlo samples to take when performing the relevant integration.

`return.alllinpred`
If `TRUE`, then the full array of predictions, on the linear predictor scale, across all MCMC samples is predicted. This is useful if the user wants to transform the predictions onto a different scale or for further manipulation, say. Defaults to `FALSE`.

`...` Not used.

Details

In the Bayesian MCMC framework, predictions are based around the posterior predictive distribution, which is the integral of the quantity one wants to predict on, integrated or averaged over the posterior distribution of the parameters and latent variables. For example, on the linear predictor scale, predictions are made as,

$$\eta_{ij} = \alpha_i + \beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j + \boldsymbol{z}_i^\top \boldsymbol{b}_j + \boldsymbol{u}_i^\top \boldsymbol{\theta}_j; \quad i = 1, \ldots, n; j = 1, \ldots, p,$$

where $\beta_{0j} + \boldsymbol{x}_i^\top \boldsymbol{\beta}_j$ is the component of the linear predictor due to the covariates $\boldsymbol{X}$ plus an intercept, $\boldsymbol{z}_i^\top \boldsymbol{b}_j$ is the component due to response-specific random intercept, $\boldsymbol{u}_i^\top \boldsymbol{\theta}_j$ is the component due to the latent variables, and α_i is the component due to one or more fixed or random row effects. Not all of these components may be included in the model, and the above is just representing the general case.

Note that for the above to work, one must have saved the MCMC samples in the fitted boral object, that is, set `save.model = TRUE` when fitting.

Two types of predictions are possible using this function:

- The first type is `predict.type = "conditional"`, meaning predictions are made conditionally on the predicted latent variables and any (random) row effects and response-specific random intercepts included in the model. This is mainly used when predictions are made onto the *same* set of sites that the model was fitted to, although a `newX` can be supplied in this case if we want to extrapolate on to the same set of sites but under different environmental conditions.
- The second type of prediction is `predict.type = "marginal"`, meaning predictions are made marginally or averaging over the latent variables and any (random) row effects and response-specific random intercepts included in the model. This is mainly used when predictions are made onto a *new* set of sites where the latent variables/row effects/response-specific random intercepts are unknown. Consequently, arguments `newX`, `newrow.ids` and `newranef.ids` are often supplied in such a setting since we are extrapolating to new observational units. The integration over the latent variables and random row effects is done via Monte-Carlo integration. Please note however that the integration is done on the linear predictor scale.

More information on conditional versus marginal predictions in latent variable models can be found in Warton et al., (2015). In both cases, and if `return.alllinpred = FALSE`, the function returns a point prediction (either the posterior mean or median depending on `est`) and the lower and upper bounds of a (100×`prob`) % interval of the posterior prediction. All of these quantities are calculated empirically based across the MCMC samples.

Value

A list containing the following components:

`linpred`	A matrix containing posterior point predictions (either posterior mean or median depending on `est`), on the linear predictor scale.
`lower`	A matrix containing the lower bound of the (100×`prob`) % interval of the posterior predictions, on the linear predictor scale.
`upper`	A matrix containing the upper bound of the (100×`prob`) % interval of the posterior predictions, on the linear predictor scale.
`all.linpred`	If `return.alllinpred = TRUE`, then only an array of predicted linear predictions across all MCMC samples.

Warnings

- Marginal predictions can take quite a while to construct due to the need to perform Monte-Carlo integration to marginalize over the latent variables and any random row effects in the model.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Gelman et al. (2013) Bayesian Data Analysis. CRC Press.
- Warton et al. (2015). So Many Variables: Joint Modeling in Community Ecology. Trends in Ecology and Evolution, 30, 766-779.

Examples

```
## Not run:
library(mvabund) ## Load a dataset from the mvabund package
library(mvtnorm)
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)

## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
```

```
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


## Example 1 - model with two latent variables, random site effects,
##  and environmental covariates
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    row.eff = "random", lv.control = list(num.lv = 2),
    mcmc.control = example_mcmc_control, save.model = TRUE,
    model.name = testpath)

## Predictions conditional on predicted latent variables
getcondpreds <- predict(spiderfit_nb)

## Predictions marginal on latent variables, random row effects
## The intervals for these will generally be wider than the
##   conditional intervals.
getmargpreds <- predict(spiderfit_nb, predict.type = "marginal")


## Now suppose you extrpolate to new sites
newX <- rmvnorm(100, mean = rep(0,ncol(X)))

## Below won't work since conditional predictions are made to the same sites
#getcondpreds <- predict(spiderfit_nb, newX = newX)

## Marginal predictions will work though, provided newrow.ids is set up
## properly. For example,
new_row_ids <- matrix(sample(1:28,100,replace=TRUE), 100, 1)
while(length(table(new_row_ids)) != 28) {
    new_row_ids <- matrix(sample(1:28,100,replace=TRUE), 100, 1)
    }
getmargpreds <- predict(spiderfit_nb, newX = newX, predict.type = "marginal",
     newrow.ids = new_row_ids)


## Example 1b - Similar to 1 except with no random site effects,
##  and a non-independence correlation structure for the latent variables
##      based on a fake distance matrix
fakedistmat <- as.matrix(dist(1:n))
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
     lv.control = list(type = "squared.exponential", num.lv = 2,
          distmat = fakedistmat), model.name = testpath,
     mcmc.control = example_mcmc_control, save.model = TRUE)

getmargpreds <- predict(spiderfit_nb, predict.type = "marginal",
     distmat = fakedistmat)

## Now suppose you extrpolate to new sites
newfakedistmat <- as.matrix(dist(1:100))
```

```
getmargpreds <- predict(spiderfit_nb, newX = newX,
     predict.type = "marginal", distmat = newfakedistmat)


## Example 1c - similar to 1 except instead of random site effects,
##   there are species-specific random intercepts at a so-called
##   "region" level
y <- spider$abun
X <- scale(spider$x)
spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    lv.control = list(num.lv = 2),
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath,
    save.model = TRUE)

## Predictions conditional on predicted latent variables and
##   random intercepts
getcondpreds <- predict(spiderfit_nb)

## Predictions marginal on latent variables, random intercepts
## The intervals for these will generally be wider than the
##   conditional intervals.
getmargpreds <- predict(spiderfit_nb, predict.type = "marginal")

## Now suppose you extrpolate to new sites
newX <- rmvnorm(100, mean = rep(0,ncol(X)))

## Marginal predictions will work though, provided newranef.ids is set up
## properly. For example,
new_ranef_ids <- matrix(sample(1:7,100,replace=TRUE), 100, 1)
getmargpreds <- predict(spiderfit_nb, newX = newX, predict.type = "marginal",
     newranef.ids = new_ranef_ids)


## Example 2 - simulate count data, based on a model with two latent variables,
## no site variables, with two traits and one environmental covariates
library(mvtnorm)

n <- 100; s <- 50
X <- as.matrix(scale(1:n))
colnames(X) <- c("elevation")

traits <- cbind(rbinom(s,1,0.5), rnorm(s))
## one categorical and one continuous variable
colnames(traits) <- c("thorns-dummy","SLA")

simfit <- list(true.lv = rmvnorm(n, mean = rep(0,2)),
 lv.coefs = cbind(rnorm(s), rmvnorm(s, mean = rep(0,2)), 1),
 traits.coefs = matrix(c(0.1,1,-0.5,0.1,0.5,0,-1,0.1), 2, byrow = TRUE))
rownames(simfit$traits.coefs) <- c("beta0","elevation")
colnames(simfit$traits.coefs) <- c("kappa0","thorns-dummy","SLA","sigma")

simy = create.life(true.lv = simfit$true.lv, lv.coefs = simfit$lv.coefs, X = X,
```

```
 traits = traits, traits.coefs = simfit$traits.coefs, family = "normal")


example_which_traits <- vector("list",ncol(X)+1)
for(i in 1:length(example_which_traits))
     example_which_traits[[i]] <- 1:ncol(traits)
fit_traits <- boral(y = simy, X = X, traits = traits,
     which.traits = example_which_traits, family = "normal",
     lv.control = list(num.lv = 2), save.model = TRUE,
     mcmc.control = example_mcmc_control,
     model.name = testpath)


## Predictions conditional on predicted latent variables
getcondpreds <- predict(fit_traits)

## Predictions marginal on latent variables
## The intervals for these will generally be wider than the
##   conditional intervals.
getmargpreds <- predict(fit_traits, predict.type = "marginal")

## End(Not run)
```

`ranefsplot` *Caterpillar plots of response-specific random effects from a fitted model*

Description

[Stable]

Constructs horizontal line plot (point estimate and HPD intervals), otherwise known as "caterpillar plots", for the response-specific random intercept predictions in the fitted model.

Usage

```
ranefsplot(sel.spp, object, ordered = FALSE, est = "median", ...)
```

Arguments

`sel.spp`	A vector selecting which response' random intercept predictions are to be plotted. It can either be a numeric vector indexing the columns of `object$y` to be plotted, or a character vector with each being an element in `colnames(object$y)` indexing the names of the responses of the plotted. Default is `NULL`, in which case plots are done for all responses, one response at a time.
`object`	An object for class "boral".

ordered	If set to TRUE, then the random intercept predictions in each caterpillar plot are plotted from smallest to largest. Defaults to FALSE, in which case the caterpillar plot is simply ordered as per the rows of object$ranef.ids.
est	A choice of either the posterior median (est = "median") or posterior mean (est = "mean"), which are then used as the point estimates in the lines. Default is posterior median.
...	Additional graphical options to be included in. These include values for cex, cex.lab, cex.axis, cex.main, lwd, and so on.

Details

For each response (column of the response matrix) and random intercept, the horizontal line or "caterpillar" is constructed by first marking the point prediction (posterior mean or median) with an "x" symbol. Then the line is construed based on the lower and upper limits of the highest posterior density (HPD) intervals as found in object$hpdintervals. By default, these are 95% HPD intervals. To complete the plot, a vertical dotted line is drawn to denote the zero value. All HPD intervals that include zero are colored gray, while HPD intervals that exclude zero are colored black.

By defaults, the plots are constructed one response at a time. That is, for a particular response, caterpillar plots of all the random intercepts (noting the number of plots is detemined by the number of random intercepts included in the model i.e., ncol(object$ranef.ids) are constructed on a single page. Then it moves onto the next response, and so on. If the user is only interested in plots from a subset of responses, then please make use of the sel.spp argument. This may be recommended especially since, in community ecology for example, the number of responses may be very large and so plotting all graphs may take a lot time.

The graph is probably better explained by, well, plotting it using the toy example below!

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

coefsplot for horizontal line or "caterpillar plot" of the regression coefficients corresponding to the covariate matrix (if applicable), the help file for ranef function in the lme4 package, for other examples of caterpillar plots of random effect predictions, caterplot from the mcmcplots package, as well as the ggpubr package, for other, sexier caterpillar plots.

Examples

```
## Not run:
library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- scale(spider$x)
n <- nrow(y)
p <- ncol(y)
```

```
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
     n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")

spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

ranefsplot(sel.spp = 1:5, object = spiderfit_nb)

ranefsplot(sel.spp = 1:5, object = spiderfit_nb, ordered = TRUE)

ranefsplot(sel.spp = c("Alopacce","Zoraspin"), object = spiderfit_nb, ordered = TRUE)

## End(Not run)
```

`summary.boral` *Summary of fitted boral object*

Description

[Stable]

A summary of the fitted boral objects including the type of model fitted e.g., error distribution, number of latent variables parameter estimates, and so on.

Usage

```
## S3 method for class 'boral'
summary(object, est = "median", ...)

## S3 method for class 'summary.boral'
print(x,...)
```

Arguments

`object`	An object of class "boral".
`x`	An object of class "boral".
`est`	A choice of either whether to print the posterior median (`est = "median"`) or posterior mean (`est = "mean"`) of the parameters.
`...`	Not used.

Value

Attributes of the model fitted, parameter estimates, and posterior probabilities of including individual and/or grouped coefficients in the model based on SSVS if appropriate.

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

See Also

boral for the fitting function on which summary is applied.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
    n.thin = 1)

testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun

spiderfit_nb <- boral(y, family = "negative.binomial",
     lv.control = list(num.lv = 2), row.eff = "fixed",
     mcmc.control = example_mcmc_control, model.name = testpath)

summary(spiderfit_nb)

## End(Not run)
```

---

tidyboral *Reformats output from a boral fit*

Description

[Maturing]

Reformats estimates from a fitted boral model to be in a slightly tidier format, meaning everything is presented as a long data frame.

Usage

```
tidyboral(object)
```

Arguments

object	An object of class "boral".

Details

Formatting the output into a long data frame maybe useful if one wishes to take the estimated parameters (namely the posterior mean/median/interquartile range/standard deviation, and the lower and upper limits of the HPD intervals), and subsequently wrangle them for presentation purposes using packages from the `tidyverse` package e.g., Wickham and Henry (2018), construct plots from them using the `ggplot2` package (Wickham, 2016), and so on.

It is important to note that this function is solely designed to format output from a fitted boral model relating to the estimated parameters. It does not an additional information like model call and MCMC samples. Please do NOT erase the original fitted model in place of this!

Value

A a list containing the following components, where applicable:

`lv.coefs`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the latent variable coefficients. This also includes the response-specific intercepts, and dispersion parameters if appropriate.
`lv`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the latent variables.
`lv.covparams`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions for the parameters characterizing the correlation structure of the latent variables when they are assumed to be non-independent across rows.
`X.coefs`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the response-specific coefficients relating to the covariate matrix.
`traits.coefs`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the coefficients and standard deviation relating to the species traits; please see `about.traits`.
`cutoffs`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the common cutoffs for ordinal responses (please see the not-so-brief tangent on distributions above).
`ordinal.sigma`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the random intercept normal distribution corresponding to the ordinal response columns.
`powerparam`	A long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the common power parameter for tweedie responses (please see the not-so-brief tangent on distributions above).
`row.coefs`	A list with each element being a long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the row effects. The length of the list is equal to the number of row effects included i.e., `object$ncol(row.ids)`.
`row.sigma`	A list with each element being a long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the row random effects normal distribution. The length of the list is equal to the number of row effects included i.e., `object$ncol(row.ids)`.

`ranef.coefs` A list with each element being a long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the response-specific random intercepts. The length of the list is equal to the number of row effects included i.e., `ncol(object$ranef.ids)`.

`ranef.sigma` A list with each element being a long format data frame containing the mean/median/standard deviation/interquartile range of the posterior distributions of the standard deviation for the response-specific random intercept normal distributions. The length of the list is equal to the number of row effects included i.e., `object$ncol(ranef.ids)`.

`ssvs.indcoefs` A long format data frame containing posterior probabilities and associated standard deviation for individual SSVS of coefficients in the covariate matrix.

`ssvs.gpcoefs` A long format data frame containing posterior probabilities and associated standard deviation for group SSVS of coefficients in the covariate matrix.

`ssvs.traitscoefs` A long format data frame containing posterior probabilities and associated standard deviation for individual SSVS of coefficients relating to species traits.

`hpdintervals` A list containing long format data frames corresponding to the lower and upper bounds of highest posterior density (HPD) intervals for all the parameters indicated above. Please see `get.hpdintervals` for more details.

Warnings

- This function is solely designed to format output from a fitted boral model relating to the estimated parameters. It does not an additional information like model call and MCMC samples. Please do NOT erase the original fitted model in place of this!

Author(s)

Francis K.C. Hui [aut, cre] (<https://orcid.org/0000-0003-0765-3533>), Wade Blanchard [aut]

Maintainer: Francis K.C. Hui <fhui28@gmail.com>

References

- Wickham, H. (2016). ggplot2: elegant graphics for data analysis. Springer.
- Wickham, H., & Henry, L. (2017). Tidyr: Easily tidy data with 'spread ()' and 'gather ()' functions.

See Also

`boral` for the fitting function on which `tidyboral` can be applied.

Examples

```
## Not run:
## NOTE: The values below MUST NOT be used in a real application;
## they are only used here to make the examples run quick!!!
example_mcmc_control <- list(n.burnin = 10, n.iteration = 100,
    n.thin = 1)
```

```
testpath <- file.path(tempdir(), "jagsboralmodel.txt")


library(mvabund) ## Load a dataset from the mvabund package
data(spider)
y <- spider$abun
X <- spider$x

spiderfit_nb <- boral(y, family = "negative.binomial",
     lv.control = list(num.lv = 2), row.eff = "fixed",
     mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb_tidy <- tidyboral(spiderfit_nb)


spiderfit_nb <- boral(y, X = X, family = "negative.binomial",
    ranef.ids = data.frame(region = rep(1:7,each=4)),
    mcmc.control = example_mcmc_control, model.name = testpath)

spiderfit_nb_tidy <- tidyboral(spiderfit_nb)

## End(Not run)
```

Index

about.distributions, 3, *19*, *24*, *34*, *38*, *41*, *54*, *72*, *76*, *85*, *91*
about.lvs, 6, *19*, *23*, *53*, *67*, *86*
about.ranefs, 8, *20*, *23, 24*, *54*, *67*, *86*, *92*
about.ssvs, 10, *15*, *21, 22*, *25*, *88*, *93*
about.traits, *11, 12*, 14, *19, 20*, *24*, *26*, *39*, *43*, *46*, *50*, *53*, *86, 87*, *91, 92*, *107*

boral, *4*, *6, 7*, *10*, *12*, *15*, 17, *35*, *38*, *42*, *54*, *56*, *63*, *67, 68*, *72*, *74*, *76–78*, *86*, *88*, *92*, *94*, *106*, *108*
boral-package, 2

calc.condlogLik, 34, *39*, *43*, *73*
calc.logLik.lv0, *36*, 37, *42, 43*
calc.marglogLik, *28*, *35, 36*, *39*, 41, *77*
calc.varpart, *29*, 45
coefsplot, *29*, 49, *104*
create.life, 51

ds.residuals, 59, *62*, *96, 97*

fitted.boral, *60*, 61, *96, 97*

get.dic, 63, *74*
get.enviro.cor, *30*, 64, *81*
get.hpdintervals, *27*, 66, *108*
get.mcmcsamples, 70
get.measures, *20*, *27*, *35*, 71, *75*, *77, 78*
get.more.measures, *63*, *74*, 75
get.residual.cor, *24*, *30*, *66*, 79

jitter, *82*, *96*

lvsplot, *23*, *30*, 82

make.jagsboralmodel, 85, *94*
make.jagsboralnullmodel, *89*, 90

plot.boral, *60*, *62*, 95
predict.boral, *30*, 97
print.boral *(*boral*)*, 17
print.summary.boral *(*summary.boral*)*, 105

ranefsplot, *9, 10*, *30*, *50*, 103

simulate.boral *(*create.life*)*, 51
summary.boral, *30*, 105

tidyboral, 106