

Package ‘because’

December 13, 2025

Type Package

Title Bayesian Estimation of Causal Effects

Version 0.9.3

Description Provides tools for specifying and fitting Bayesian Structural Equation Models using JAGS. The 'because' package allows causal inference in complex ecological and evolutionary analyses, accounting for phylogenetic uncertainty, spatial autocorrelation, and other covariance structures. Implements and expands on the methods described in von Hardenberg and Gonzalez-Voyer (2025) <doi:10.1111/2041-210X.70044>.

License AGPL-3

Encoding UTF-8

LazyData true

Imports rjags,
ape,
ggm,
coda,
stats,
parallel,
utils

Depends R (>= 3.5)

RoxygenNote 7.3.3

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0),
devtools,
MASS,
loo

VignetteBuilder knitr, rmarkdown

Contents

because	2
because_compare	5
because_dsep	7
because_format_data	8
because_lambda	9
because_loo	10

because_loo_compare	11
because_model	12
because_waic	14
print.summary.because	15
rhino.dat	16
rhino.tree	17
storks	17
summary.because	18

Index	19
--------------	-----------

because	<i>Run a Phylogenetic Bayesian Structural Equation model (Because)</i>
---------	------------------------------------------------------------------------

Description

Run a Phylogenetic Bayesian Structural Equation model (Because)

Usage

```
because(
  data,
  equations,
  id_col = NULL,
  structure = NULL,
  tree = NULL,
  monitor = NULL,
  n.chains = 3,
  n.iter = 12500,
  n.burnin = n.iter/5,
  n.thin = 10,
  DIC = TRUE,
  WAIC = FALSE,
  n.adapt = n.iter/5,
  quiet = FALSE,
  dsep = FALSE,
  variability = NULL,
  distribution = NULL,
  latent = NULL,
  latent_method = c("correlations", "explicit"),
  standardize_latent = TRUE,
  parallel = FALSE,
  n.cores = 1,
  cl = NULL,
  ic_recompile = TRUE,
  optimise = TRUE,
  random = NULL,
  levels = NULL,
  hierarchy = NULL,
  link_vars = NULL,
  fix_residual_variance = NULL
)
```

Arguments

<code>data</code>	<p>Data for the model. Accepts:</p> <ul style="list-style-type: none"> • <code>data.frame</code>: A data frame with variables as columns. Variables needed for the model are automatically extracted from the equations. Extra columns are ignored. • <code>list</code>: A named list where each element is a vector of values (traditional format for backward compatibility).
<code>equations</code>	A list of model formulas describing the structural equation model.
<code>id_col</code>	<p>Character string specifying the column name in a <code>data.frame</code> containing unit identifiers (species, individuals, sites, etc.). This is used to:</p> <ul style="list-style-type: none"> • Match data rows to tree tip labels (for phylogenetic models) • Link data to external spatial or custom covariance matrices. <p>**Note**: For standard random effects models (e.g. <code>random = ~(1 species)</code>) where no external structure (like a tree) is provided, this argument is **not required**. The grouping column is read directly from the data.</p> <p>If <code>NULL</code> (default): uses meaningful row names if available. Ignored when data is already a list.</p>
<code>structure</code>	<p>The covariance structure for the model. Accepts:</p> <ul style="list-style-type: none"> • <code>"phylo"</code> object: Phylogenetic tree (Standard PGLS/PhyloSEM). • <code>"multiPhylo"</code> object: List of trees (incorporates phylogenetic uncertainty). • <code>NULL</code>: Independent model (Standard SEM, no covariance structure). • <code>matrix</code>: Custom covariance or precision matrix (e.g., spatial connectivity, kinship).
<code>tree</code>	(Deprecated alias for <code>structure</code>). A single phylogenetic tree of class <code>"phylo"</code> or a list of trees. Use <code>structure</code> instead for new code.
<code>monitor</code>	<p>Parameter monitoring mode. Options:</p> <ul style="list-style-type: none"> • <code>"interpretable"</code> (default): Monitor only scientifically meaningful parameters: intercepts (alpha), regression coefficients (beta), phylogenetic signals (lambda) for responses, and WAIC terms. Excludes variance components (tau) and auxiliary predictor parameters. • <code>"all"</code>: Monitor all model parameters including variance components and implicit equation parameters. • Custom vector: Provide a character vector of specific parameter names to monitor. • <code>NULL</code>: Auto-detect based on model structure (equivalent to <code>"interpretable"</code>).
<code>n.chains</code>	Number of MCMC chains (default = 3).
<code>n.iter</code>	Total number of MCMC iterations (default = 12500).
<code>n.burnin</code>	Number of burn-in iterations (default = <code>n.iter / 5</code>).
<code>n.thin</code>	Thinning rate (default = 10).
<code>DIC</code>	<p>Logical; whether to compute DIC using <code>dic.samples()</code> (default = <code>TRUE</code>).</p> <p>**Note**: DIC penalty will be inflated for models with measurement error or repeated measures because latent variables are counted as parameters (penalty ~ structural parameters + N). For model comparison, use WAIC or compare mean deviance across models with similar structure.</p>
<code>WAIC</code>	<p>Logical; whether to sample values for WAIC and deviance (default = <code>FALSE</code>).</p> <p>WAIC is generally more appropriate than DIC for hierarchical models with latent variables.</p>

n.adapt	Number of adaptation iterations (default = $n.iter / 5$).
quiet	Logical; suppress JAGS output (default = FALSE).
dsep	Logical; if TRUE, monitor only the first beta in each structural equation (used for d-separation testing).
variability	<p>Optional specification for variables with measurement error or within-species variability. **AUTO-DETECTION**: If a variable X has a corresponding column X_se (standard errors), X_obs (repeated measures), or is provided as a matrix, variability is automatically detected.</p> <p>**Manual specification** (for non-standard column names):</p> <ul style="list-style-type: none"> • Simple: <code>list(X = "se", Y = "reps")</code> - uses standard X_se/Y_obs naming • Custom columns: <code>list(X = list(type = "se", se_col = "X_SD"))</code> - specify custom column names • For SE: <code>se_col</code> (SE column), <code>mean_col</code> (mean column, optional) • For reps: <code>obs_col</code> (observations matrix column)
distribution	<p>Optional named character vector specifying the distribution for response variables. Default is "gaussian" for all variables. Supported values:</p> <ul style="list-style-type: none"> • "gaussian" (default) • "binomial" (binary data) • "multinomial" (unordered categorical > 2 levels) • "ordinal" (ordered categorical > 2 levels) • "poisson" (count data) • "negbinomial" (overdispersed count data) • "zip" (zero-inflated poisson): Models excess zeros with probability ψ and counts with mean λ. • "zinb" (zero-inflated negative binomial): Models excess zeros with probability ψ and overdispersed counts with mean μ and size r. <p>The model will estimate a zero-inflation probability parameter ψ for these distributions. Example: <code>distribution = c(Gregarious = "binomial")</code>.</p>
latent	Optional character vector of latent (unmeasured) variable names. If specified, the model will account for induced correlations among observed variables that share these latent common causes.
latent_method	<p>Method for handling latent variables (default = "correlations").</p> <ul style="list-style-type: none"> • "correlations": MAG approach - marginalize latent variables and estimate induced correlations (ρ) between observed variables that share latent parents. • "explicit": Model latent variables as JAGS nodes and estimate structural paths from latents to observed variables.
standardize_latent	Logical; if TRUE and <code>latent_method = "explicit"</code> , adds standardized priors ($N(0, 1)$) to latent variables to identify scale and location. This improves convergence and makes regression coefficients interpretable as standardized effects. Only applicable when using explicit latent variable modeling (default = TRUE).
parallel	Logical; if TRUE, run MCMC chains in parallel (default = FALSE). Note: Requires <code>n.cores > 1</code> to take effect.
n.cores	Integer; number of CPU cores to use for parallel chains (default = 1). Only used when <code>parallel = TRUE</code> .

<code>cl</code>	Optional; a cluster object created by <code>parallel::makeCluster()</code> . If <code>NULL</code> , a cluster will be created and destroyed automatically.
<code>ic_recompile</code>	Logical; if <code>TRUE</code> and <code>parallel = TRUE</code> , recompile the model after parallel chains to compute DIC/WAIC (default = <code>TRUE</code>). This adds a small sequential overhead but enables information criteria calculation.
<code>optimise</code>	Logical; if <code>TRUE</code> (default), use the optimized random effects formulation for phylogenetic models. This is significantly faster (5-10x) and more numerically stable. If <code>FALSE</code> , use the traditional marginal formulation (slower, but provided for comparison).
<code>random</code>	Optional formula or list of formulas specifying global random effects applied to all equations (e.g. <code>~(1 species)</code>).
<code>levels</code>	(Hierarchical Data) A named list mapping variables to their hierarchy levels. Required if data is a list of data frames (hierarchical format). Example: <code>list(individual = c("y", "x"), site = c("z"))</code> .
<code>hierarchy</code>	(Hierarchical Data) Character string describing the topological ordering of levels (e.g., <code>"site > individual"</code>). Required for hierarchical data if not fully inferred from random effects.
<code>link_vars</code>	(Hierarchical Data) Optional named character vector specifying variables used to link data levels (e.g. <code>c(site = "site_id")</code>).
<code>fix_residual_variance</code>	Optional named vector for fixing residual variances. Useful for handling non-identified models or specific theoretical constraints. Example: <code>c(response_var = 1)</code> .

Value

A list of class "because" with model output and diagnostics.

because_compare	<i>Compare Because Models</i>
-----------------	-------------------------------

Description

A unified function to either (1) compare previously fitted models, or (2) run multiple model specifications in parallel and then compare them.

Usage

```
because_compare(
  ...,
  model_specs = NULL,
  data = NULL,
  tree = NULL,
  n.cores = 1,
  cl = NULL,
  sort = TRUE
)
```

Arguments

<code>...</code>	For comparing fitted models: individual fitted model objects of class "because". For running models: additional arguments passed to <code>because</code> (e.g., <code>n.iter</code>).
<code>model_specs</code>	A named list of model specifications to run (Mode 2). Each element should be a list containing arguments for <code>because</code> . Alternatively, this argument can accept the first fitted model object (Mode 1).
<code>data</code>	The dataset (required for Mode 2). Alternatively, the second fitted model object (Mode 1).
<code>tree</code>	The phylogenetic tree (optional for Mode 2). Alternatively, the third fitted model object (Mode 1).
<code>n.cores</code>	Number of cores for parallel execution (Mode 2). Default is 1.
<code>cl</code>	Optional cluster object (Mode 2).
<code>sort</code>	Logical. If TRUE (default), sort comparison table by WAIC.

Details

Mode 1: Compare Fitted Models Call `because_compare(fit1, fit2)` or `because_compare(models = list(fit1, fit2))`. Extracts WAIC (with SE) from each model and ranks them.

Mode 2: Run and Compare Call `because_compare(model_specs = list(m1=..., m2=...), data=data, tree=tree)`. This runs the models in parallel and returns the comparison.

Value

If comparing fitted models: A class "because_comparison" object (data frame) with WAIC rankings.

If running models: A list containing:

<code>results</code>	List of fitted model objects.
<code>comparison</code>	The comparison data frame.

Examples

```
## Not run:
# Mode 1: Compare existing fits
because_compare(fit1, fit2)

# Mode 2: Run and compare
specs <- list(m1 = list(equations = list(Y ~ X)), m2 = list(equations = list(Y ~ X + Z)))
res <- because_compare(specs, data = df, tree = tr, n.cores = 2)
print(res$comparison)

## End(Not run)
```

because_dsep	<i>Extract d-separation statements from a structural equation model</i>
--------------	-------------------------------------------------------------------------

Description

This function takes a set of structural equations defining a causal model and returns the conditional independence statements (d-separation or m-separation tests) implied by the model structure. If latent variables are specified, the function uses the MAG (Maximal Ancestral Graph) approach by Shipley and Douma (2021) to account for unmeasured latent variables.

Usage

```
because_dsep(
  equations,
  latent = NULL,
  random_terms = list(),
  hierarchical_info = NULL,
  quiet = FALSE
)
```

Arguments

equations	A list of model formulas (one per structural equation), e.g., <code>list(Y ~ X1 + X2, Z ~ Y)</code> .
latent	Optional character vector of latent (unmeasured) variable names. If provided, the function converts the DAG to a MAG and returns m-separation tests.
random_terms	Optional list of random effects (group, type) parsed from equations.
hierarchical_info	Internal argument used to pass data hierarchy information (levels, grouping variables) for future implementation of multilevel d-separation tests (following Shipley 2009). Currently unused by the d-separation logic.
quiet	Logical; if FALSE (default), print the basis set and MAG structure. If TRUE, suppress informational output.

Details

The function implements the basis set approach to d-separation testing (Shipley 2000, 2009, 2016). For standard DAGs without latent variables, it identifies pairs of non-adjacent variables and creates conditional independence tests.

When latent variables are specified, the function uses the DAG-to-MAG conversion (Shipley & Douma 2021) to identify m-separation statements and induced correlations among observed variables that arise from shared latent common causes.

Value

If `latent` is `NULL`, returns a list of formulas representing conditional independence tests. If `latent` is specified, returns a list with:

- `tests`: List of m-separation test formulas
- `correlations`: List of variable pairs with induced correlations

References

- Shipley, B. (2000). A new inferential test for path models based on directed acyclic graphs. *Structural Equation Modeling*, 7(2), 206-218.
- Shipley, B. (2009). Confirmatory path analysis in a generalized multilevel context. *Ecology*, 90(2), 363-368.
- Shipley, B. (2016). *Cause and Correlation in Biology* (2nd ed.). Cambridge University Press.
- Shipley, B., & Douma, J. C. (2021). Testing Piecewise Structural Equations Models in the Presence of Latent Variables and Including Correlated Errors. *Structural Equation Modeling: A Multidisciplinary Journal*, 28(4), 582–589. <https://doi.org/10.1080/10705511.2020.1871355>

Examples

```
# Standard DAG
equations <- list(LS ~ BM, NL ~ BM + RS, DD ~ NL)
ind_tests <- because_dsep(equations)

# With latent variable
equations_latent <- list(X ~ Quality, Y ~ Quality)
result <- because_dsep(equations_latent, latent = "Quality")
# result$tests: m-separation tests
# result$correlations: induced correlation between X and Y
```

because_format_data *Format Data for Because Analysis*

Description

Converts data from long format (one row per observation) to the list format required by [because](#).

Usage

```
because_format_data(data, species_col = "SP", tree)
```

Arguments

data	A data.frame in long format with one row per observation.
species_col	Name of the column containing species identifiers (default: "SP").
tree	A phylogenetic tree (class phylo). Required to determine species order.

Details

This function handles:

- Different numbers of replicates per species (creates rectangular matrix with NA padding)
- Missing values (NA)
- Automatic alignment with phylogenetic tree tip labels

When species have different numbers of replicates, the function creates a matrix with dimensions (number of species) x (maximum number of replicates). Species with fewer replicates are padded with NA values.

Species in the tree but not in the data will have all NA values. Species in the data but not in the tree will be excluded with a warning.

Value

A named list where each element is either:

- A numeric vector (if all species have exactly 1 observation)
- A numeric matrix with species in rows and replicates in columns

Species are ordered to match `tree$tip.label`.

Examples

```
## Not run:
# Example data in long format
data_long <- data.frame(
  SP = c("sp1", "sp1", "sp1", "sp2", "sp2", "sp3"),
  BM = c(1.2, 1.3, 1.1, 2.1, 2.2, 1.8),
  NL = c(0.5, 0.6, NA, 0.7, 0.8, 0.9)
)

tree <- ape::read.tree(text = "(sp1:1,sp2:1,sp3:1);")
data_list <- because_format_data(data_long, species_col = "SP", tree = tree)

# Use with because
fit <- because(data = data_list, tree = tree, equations = list(NL ~ BM))

## End(Not run)
```

because_lambda

Calculate Pagel's Lambda from Variance Components

Description

This function calculates Pagel's lambda for each response variable in a fitted because model by extracting the posterior samples of the phylogenetic and residual variance components.

Usage

```
because_lambda(model, prob = 0.95)
```

Arguments

<code>model</code>	A fitted model object of class "because".
<code>prob</code>	A numeric value specifying the probability mass for the credibility intervals (default 0.95).

Details

In the optimized formulation of because, the phylogenetic signal (λ) is not always explicitly monitored for complex models with multiple structures. However, it can be derived post-hoc from the estimated standard deviations of the phylogenetic (σ_{phylo}) and residual (σ_{res}) components:

$$\lambda = \frac{\sigma_{phylo}^2}{\sigma_{phylo}^2 + \sigma_{res}^2}$$

This function performs this calculation on the full posterior chains to provide valid Bayesian estimates and credibility intervals.

Value

A data frame containing the summary statistics for the derived lambda parameter(s):

Mean	Posterior mean
SD	Posterior standard deviation
Median	Posterior median
LowerCI	Lower bound of the credibility interval
UpperCI	Upper bound of the credibility interval

Examples

```
## Not run:
fit <- because(...)
because_lambda(fit)

## End(Not run)
```

because_loo

Calculate LOO-CV for a Because Model

Description

Calculates Leave-One-Out Cross-Validation using Pareto Smoothed Importance Sampling (PSIS-LOO) for a fitted Because model.

Usage

```
because_loo(model, ...)
```

Arguments

model	A fitted model object of class "because" returned by because .
...	Additional arguments passed to <code>loo::loo()</code> .

Details

LOO-CV (Leave-One-Out Cross-Validation) uses Pareto Smoothed Importance Sampling to approximate leave-one-out predictive performance without refitting the model N times. This is particularly useful for:

- Model comparison when models have different numbers of latent variables
- Identifying influential observations (via Pareto k diagnostics)
- Robust predictive performance assessment

****Pareto k diagnostics**:**

- $k < 0.5$: Excellent (all estimates reliable)

- $0.5 < k < 0.7$: Good (estimates okay)
- $0.7 < k < 1$: Problematic (estimates unreliable)
- $k > 1$: Very problematic (refit model excluding these observations)

****Note on implementation****: This function computes pointwise log-likelihoods from the posterior samples. For phylogenetic models, this is done by evaluating the multivariate normal log-likelihood for each observation given the posterior parameter values.

Value

A loo object containing:

estimates	Table with ELPD (expected log pointwise predictive density), LOO-IC, and p_loo
diagnostics	Pareto k diagnostic values for each observation
pointwise	Pointwise contributions to LOO-IC

Examples

```
## Not run:
fit <- because(data, tree, equations)
loo_result <- because_loo(fit)
print(loo_result)

# Check for problematic observations
plot(loo_result)

# Compare models
loo_compare(loo_result1, loo_result2)

## End(Not run)
```

because_loo_compare	<i>Compare Models Using LOO-CV</i>
---------------------	------------------------------------

Description

Wrapper for `loo::loo_compare()` to compare multiple Because models.

Usage

```
because_loo_compare(...)
```

Arguments

... Two or more loo objects from `because_loo()`.

Value

A comparison table ranking models by expected out-of-sample predictive accuracy.

Examples

```
## Not run:
  loo1 <- because_loo(fit1)
  loo2 <- because_loo(fit2)
  because_loo_compare(loo1, loo2)

## End(Not run)
```

because_model	<i>Generate a JAGS model string for Phylogenetic Bayesian SEM (Because)</i>
---------------	-----------------------------------------------------------------------------

Description

This function builds the model code to be passed to JAGS based on a set of structural equations. It supports both single and multiple phylogenetic trees (to account for phylogenetic uncertainty). Missing values are handled both in the response and predictor variables treating all of them as stochastic nodes.

Usage

```
because_model(
  equations,
  multi.tree = FALSE,
  latent_method = "correlations",
  structure_names = "phylo",
  random_structure_names = NULL,
  random_terms = list(),
  vars_with_na = NULL,
  induced_correlations = NULL,
  variability = NULL,
  distribution = NULL,
  optimise = TRUE,
  standardize_latent = TRUE,
  poly_terms = NULL,
  latent = NULL,
  compute_waic = FALSE,
  fix_residual_variance = NULL
)
```

Arguments

equations	A list of model formulas.
multi.tree	Logical; if TRUE, incorporates phylogenetic uncertainty by sampling across a set of trees.
structure_names	(Internal) Character vector of names for multiple trees/structures.
vars_with_na	Optional character vector of response variable names that have missing data. These variables will use element-wise likelihoods instead of multivariate normal.

induced_correlations	Optional list of variable pairs with induced correlations from latent variables. Each element should be a character vector of length 2 specifying the pair of variables that share a latent common cause.
variability	Optional character vector or named character vector of variable names that have measurement error or within-species variability. If named, the names should be the variable names and the values should be the type of variability: "se" (for mean and standard error) or "reps" (for repeated measures). If unnamed, it defaults to "se" for all specified variables. <ul style="list-style-type: none"> "se": Expects Var_mean and Var_se in the data. The model fixes observation error: $\text{Var_mean} \sim \text{dnorm}(\text{Var}, 1/\text{Var_se}^2)$. "reps": Expects Var_obs (matrix) and N_reps_Var (vector) in the data. The model estimates observation error: $\text{Var_obs}[i, j] \sim \text{dnorm}(\text{Var}[i], \text{Var_tau})$.
distribution	Optional named character vector specifying the distribution for response variables. Default is "gaussian" for all variables. Supported values: "gaussian", "binomial", "multinomial". For "binomial" variables, the model uses a logit link and a Bernoulli likelihood, with phylogenetic correlation modeled on the latent scale.
optimise	Logical. If TRUE (default), use random effects formulation for 4.6× speedup. If FALSE, use original marginal covariance formulation.
standardize_latent	Logical (default TRUE). If TRUE, standardizes latent variables to unit variance.
poly_terms	(Internal) List of polynomial terms for model generation.
latent	Optional character vector of latent variable names.
fix_residual_variance	Optional numeric value or named vector to fix residual variance.

Details

The generated model includes:

- Linear predictors and multivariate normal likelihoods for each response variable.
- Priors for intercepts (alpha), slopes (beta), lambda parameters (lambda), and residual precisions (tau).
- Phylogenetic covariance modeled via a single VCV matrix (when `multi.tree = FALSE`) or a 3D array `multiVCV[, , K]` with categorical sampling across trees (when `multi.tree = TRUE`).
- (Optional) Observation models for variables with measurement error:
 - Type "se": $\text{Var_mean} \sim \text{dnorm}(\text{Var}, 1/\text{Var_se}^2)$
 - Type "reps": $\text{Var_obs}[i, j] \sim \text{dnorm}(\text{Var}[i], \text{Var_tau})$
- (Optional) Generalized linear mixed models for non-Gaussian responses (e.g., binomial).
- (Optional) Element-wise likelihoods for response variables with missing data.

Value

A list with two elements:

- `model`: A character string containing the JAGS model code.
- `parameter_map`: A data frame mapping response variables to their predictors and parameter names.

Examples

```
eqs <- list(BR ~ BM, S ~ BR, G ~ BR, L ~ BR)
cat(because_model(eqs, multi.tree = TRUE)$model)
```

because_waic

Calculate WAIC with Standard Errors for a Because Model

Description

Calculates the Widely Applicable Information Criterion (WAIC) with standard errors for a fitted Because model using pointwise log-likelihoods.

Usage

```
because_waic(model)
```

Arguments

model A fitted model object of class "because" returned by `because` with `WAIC = TRUE`.

Details

This function implements WAIC following Vehtari et al. (2017). It requires that the model was run with `WAIC = TRUE` to monitor pointwise log-likelihoods.

Value

A data frame with columns `Estimate` and `SE` containing:

<code>elpd_waic</code>	Expected log pointwise predictive density (higher is better)
<code>p_waic</code>	Effective number of parameters
<code>waic</code>	The WAIC value (lower is better for model comparison)

The returned object also has a pointwise attribute containing individual observation contributions for model comparison.

WAIC Definition

The Widely Applicable Information Criterion (WAIC) is calculated as:

$$WAIC = -2 \times (lppd - p_{waic})$$

where:

- $lppd = \sum_{i=1}^N \log(\frac{1}{S} \sum_{s=1}^S \exp(\log_{lik_{is}}))$ is the log pointwise predictive density
- $p_{waic} = \sum_{i=1}^N \text{var}(\log_{lik_{is}})$ is the effective number of parameters

****WAIC Algorithm**:**

1. **lpd** (log pointwise predictive density): For each observation i , compute $\log(\text{mean}(\exp(\log_{lik_i})))$ across MCMC samples
2. **p_waic**: For each observation i , compute $\text{var}(\log_{lik_i})$ across MCMC samples
3. **elpd_waic**: $lpd_i - p_{waic_i}$ for each observation
4. **waic**: $-2 \times \sum \text{elpd_waic}_i$

Standard Errors

Standard errors for WAIC are calculated using the pointwise contributions:

$$SE(WAIC) = \sqrt{N \times \text{var}(waic_i)}$$

where $waic_i = -2 \times (lppd_i - p_{waic,i})$.

References

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5), 1413-1432.

Examples

```
## Not run:
# Fit model with WAIC monitoring
fit <- because(data, tree, equations, WAIC = TRUE)

# View WAIC with standard errors
fit$WAIC
#           Estimate    SE
# elpd_waic   -617.3   12.4
# p_waic       12.3    3.1
# waic        1234.5   24.8

# Compare two models
fit1$WAIC
fit2$WAIC
# Model with lower WAIC is preferred
# Difference is significant if |WAIC1 - WAIC2| > 2 * sqrt(SE1^2 + SE2^2)

## End(Not run)
```

print.summary.because *Print Summary for Because Model*

Description

Print Summary for Because Model

Usage

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

Arguments

x A summary object of class "summary.because".
 ... Additional arguments.

rhino.dat

*Rhinograd life-history data***Description**

A dataset containing simulated life-history trait data for 100 Rhinograd species. This data is used to demonstrate phylogenetic Bayesian structural equation modeling with the becauseR package.

Usage

```
rhino.dat
```

Format

A data frame with 100 rows (one per species) and 6 columns:

SP Character. Species names matching the tip labels in rhino.tree

BM Numeric. Body mass (standardised)

LS Numeric. Litter size (standardised)

NL Numeric. Nose length (standardised)

DD Numeric. Dispersal distance (standardised)

RS Numeric. Range size (standardised)

Details

This simulated dataset represents life-history traits for hypothetical Rhinograd species. The data can be used to test causal hypotheses about life-history evolution using phylogenetic structural equation models.

Example causal model (Model 8 from Gonzalez-Voyer and von Hardenberg (2014)):

- Body mass (BM) affects litter size (LS)
- Body mass (BM) and range size (RS) affect nose length (NL)
- Nose length (NL) affects dispersal distance (DD)

Source

Simulated data for Gonzalez-Voyer & von Hardenberg (2014)

References

Gonzalez-Voyer, A., & von Hardenberg, A. (2014). An introduction to phylogenetic path analysis. In: Modern phylogenetic comparative methods and their application in evolutionary biology (pp. 201–229). Springer.

Examples

```
data(rhino.dat)
head(rhino.dat)
summary(rhino.dat)
```

rhino.tree	<i>Rhinograd phylogenetic tree</i>
------------	------------------------------------

Description

A phylogenetic tree (phylo object) for 100 simulated Rhinograd species. This tree is used to demonstrate phylogenetic Bayesian structural equation modeling with the becauseR package.

Usage

```
rhino.tree
```

Format

A phylo object (from the ape package) with 100 tips and 99 internal nodes. The tree has been scaled so that the root age is 1.0.

Details

Rhinograds are hypothetical mammals used as examples in phylogenetic comparative methods. This simulated tree represents the evolutionary relationships among 100 Rhinograd species.

Source

Simulated data for Gonzalez-Voyer and von Hardenberg (2014)

References

Gonzalez-Voyer, A., & von Hardenberg, A. (2014). An introduction to phylogenetic path analysis. In: Modern phylogenetic comparative methods and their application in evolutionary biology (pp. 201–229). Springer.

Examples

```
data(rhino.tree)
plot(rhino.tree)
```

storks	<i>Data on Stork Populations and Human Birth Rates</i>
--------	--------------------------------------------------------

Description

A dataset containing information on stork populations, human birth rates, and area size for 17 European countries, as described in Matthews (2000). This dataset is famously used to demonstrate spurious correlations (d-separation).

Usage

```
storks
```

Format

- A data frame with 17 rows and 5 variables:
- Country** Name of the country
- Area** Area of the country in square kilometers
- Storks** Number of breeding stork pairs
- Humans** Human population size (in millions)
- Birth** Human birth rate (1000s per year)

Source

Matthews, R. (2000). Storks deliver babies ($p=0.008$). *Teaching Statistics*, 22(2), 36-38.

summary.because	<i>Summary for Because Model</i>
-----------------	----------------------------------

Description

Summarizes the output of a Because model run.

Usage

```
## S3 method for class 'because'  
summary(object, ...)
```

Arguments

- object A fitted model object of class "because".
- ... Additional arguments passed to [summary.mcmc](#).

Value

A summary object containing statistics for the monitored parameters. If dsep = TRUE was used in because, the summary focuses on the conditional independence tests.

Index

* datasets

- rhino.dat, [16](#)
- rhino.tree, [17](#)
- storks, [17](#)

- because, [2](#), [6](#), [8](#), [10](#), [14](#)
- because_compare, [5](#)
- because_dsep, [7](#)
- because_format_data, [8](#)
- because_lambda, [9](#)
- because_loo, [10](#)
- because_loo_compare, [11](#)
- because_model, [12](#)
- because_waic, [14](#)

- print.summary.because, [15](#)

- rhino.dat, [16](#)
- rhino.tree, [17](#)

- storks, [17](#)
- summary.because, [18](#)
- summary.mcmc, [18](#)