

Package: coevolve (via r-universe)

October 7, 2026

Title Fit Bayesian Generalized Dynamic Phylogenetic Models using
'Stan' or 'JAX'

Version 1.2.1

Description Fit Bayesian generalized dynamic phylogenetic models using
'Stan' or 'JAX'. The package allows an arbitrary number of
variables with different response distributions to coevolve on
a phylogenetic tree via an Ornstein-Uhlenbeck evolutionary
process, allowing users to assess directionality between
coevolving variables. The optional JAX backend enables
high-performance sampling with NumPyro and potential GPU
acceleration via 'reticulate'.

License GPL (>= 3)

Date 2026-06-23

URL <https://github.com/ScottClaessens/coevolve>,
<https://scottclaessens.github.io/coevolve/>

BugReports <https://github.com/ScottClaessens/coevolve/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c("` namespace", "` rd",
` srr::srr_stats_roclet"))

Suggests knitr, loo, rmarkdown, reticulate, testthat (>= 3.1.7),
tibble, withr

Config/testthat/edition 3

Imports ape, bayesplot, cmdstanr, colorspace, dplyr, ggplot2,
gridExtra, lifecycle, Matrix, methods, patchwork (>= 1.2.0),
phangorn, phaseR, phytools, posterior, purrr, readr, rlang,
stats, stringr, tidyr, whisker

VignetteBuilder knitr

Depends R (>= 4.1.0)

LazyData true

Additional_repositories <https://stan-dev.r-universe.dev/>

Remotes stan-dev/cmdstanr, mjb211/phaseR
Config/roxygen2/version 8.0.0
Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev libx11-dev
Repository <https://ropensci-staging.r-universe.dev>
Date/Publication 2026-10-06 19:31:38 UTC
RemoteUrl <https://github.com/ScottClaessens/coevolve>
RemoteRef HEAD
RemoteSha 52bdfcc1cf47a5f53adb7a181e2449318b524430

Contents

authority	3
coev_ancestral_states	4
coev_calculate_delta_theta	6
coev_calculate_theta	8
coev_fit	10
coev_make_stancode	15
coev_make_standata	20
coev_plot_delta_theta	24
coev_plot_flowfield	26
coev_plot_pred_series	28
coev_plot_predictive_check	29
coev_plot_selection_gradient	31
coev_plot_trait_values	33
coev_pred_series	34
coev_simulate_coevolution	36
draws.jax_fit	39
draws.nutpie_fit	39
extract_samples	40
metadata.jax_fit	41
metadata.nutpie_fit	41
pairs.coevfit	42
plot.coevfit	43
primates	44
print.coevfit	46
repeated	47
save_coevfit	48
stancode	49
standata	50
summary.coevfit	51
summary.jax_fit	52
summary.nutpie_fit	53

Index	54
--------------	-----------

authority

Political and religious authority in Austronesian societies

Description

This dataset and associated phylogeny describes the states of political and religious authority in 97 Austronesian societies. These data were originally compiled in Sheehan et al. (2023). Political authority is defined as a right to manage interactions between living human beings, whereas religious authority is defined as a right to manage interactions between living human beings and supernatural agents or powers. Authority is coded as being absent, sublocal (smaller than the local community), local (coextensive with the local community), or supralocal (consisting of more than one local community).

Usage

authority

Format

A list containing a dataset, an associated language phylogeny, and associated geographic coordinates. The phylogeny is a pruned maximum clade credibility tree for 97 Austronesian languages. The coordinates are in a data frame, with longitude and latitude values measured in decimal degrees. The dataset is a data frame with 97 observations and three variables:

language The name of the Austronesian language linked to each society

political_authority An ordered factor indicating whether political authority is absent, sublocal, local, or supralocal in each society

religious_authority An ordered factor indicating whether religious authority is absent, sublocal, local, or supralocal in each society

Source

Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45.

Examples

```
## Not run:
# fit model to authority data
m <-
  coev_fit(
    data = authority$data,
    variables = list(
      political_authority = "ordered_logistic",
      religious_authority = "ordered_logistic"
    ),
    id = "language",
```

```

    tree = authority$phylogeny,
    # arguments for cmdstanr::sample()
    parallel_chains = 4,
    seed = 1,
    # set prior manually
    prior = list(A_offdiag = "normal(0, 2)")
  )
# print model summary
summary(m)
# plot delta theta
coev_plot_delta_theta(m)

## End(Not run)

```

coev_ancestral_states *Extract ancestral state estimates from a fitted coevolutionary model*

Description

Extracts posterior estimates of trait values at internal nodes of the phylogenetic tree from a fitted coevfit object. By default, estimates are returned on the latent (eta) scale; use `scale = "response"` to apply the inverse link for each variable's response distribution. The latent eta parameters in the model correspond directly to nodes in the phylogeny, making ancestral state reconstruction a natural byproduct of model fitting.

Usage

```

coev_ancestral_states(
  object,
  variables = NULL,
  nodes = "internal",
  tree_id = NULL,
  scale = "latent",
  summary = TRUE,
  prob = 0.95
)

```

Arguments

<code>object</code>	An object of class <code>coevfit</code>
<code>variables</code>	(optional) A character vector of variable names to include. If <code>NULL</code> (default), all variables from the fitted model are included.
<code>nodes</code>	Which nodes to include. Either <code>"internal"</code> (default) for internal nodes only, <code>"all"</code> for both tips and internal nodes, or an integer vector of specific node IDs (using ape's node numbering: tips = 1:N_tips, internal = (N_tips+1):total).

tree_id	(optional) An integer indicating which tree to use when the model was fit with a multiPhylo object. If NULL (default), uses tree 1. (Integration over trees with topological uncertainty is not yet implemented.)
scale	Character string, either "latent" (default) to return values on the latent eta scale, or "response" to apply the inverse link function for each variable's response distribution.
summary	Logical. If TRUE (default), returns a long-format data frame with posterior point estimates and credible intervals. If FALSE, returns the raw posterior draws.
prob	A single numeric value between 0 and 1 indicating the desired probability mass for the credible interval. Default is 0.95.

Value

If summary = TRUE, a long-format data frame (tibble) with columns:

node Integer node ID in the reference tree

variable Character variable name

category Only present when scale = "response". For ordinal variables, the category label ("cat_1", "cat_2", ...). NA for non-ordinal variables.

estimate Posterior median

lower Lower bound of the credible interval

upper Upper bound of the credible interval

The data frame has attributes ref_tree (the phylo object used), prob, and scale.

If summary = FALSE, a list with elements:

draws A 3D array (draws x nodes x variables)

ref_tree The reference phylo object

node_ids Integer vector of node IDs (matching dim 2 of draws)

variable_names Character vector of variable names

See Also

[coev_fit](#), [coev_plot_trait_values](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
```

```

    chains = 4,
    parallel_chains = 4,
    seed = 1
  )

  # extract ancestral states (latent scale, internal nodes, summary)
  asr <- coev_ancestral_states(fit)

  # extract all nodes including tips
  asr_all <- coev_ancestral_states(fit, nodes = "all")

  # extract on response scale
  asr_resp <- coev_ancestral_states(fit, scale = "response")

  # get raw posterior draws
  asr_draws <- coev_ancestral_states(fit, summary = FALSE)

  ## End(Not run)

```

coev_calculate_delta_theta

Calculate delta theta from a fitted coevfit object

Description

Calculate $\Delta\theta$ from a fitted coevfit object. $\Delta\theta$ is defined as the difference in the equilibrium value for a "response" trait that results from a standardised increase in the value of another "predictor" trait. This value can be used to assess contingencies and directionality between variables in the coevolutionary process.

Usage

```
coev_calculate_delta_theta(object, response, predictor)
```

Arguments

object	An object of class coevfit
response	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)
predictor	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)

Details

This function calculates $\Delta\theta$, which is defined as the difference in the equilibrium value for a "response" trait that results from a standardised increase in the value of another "predictor" trait. The function first calculates the equilibrium trait value for the response trait when the predictor trait is

held at its empirical median value (the [coev_calculate_theta](#) function is used for this purpose, see `help(coev_calculate_theta)` for further details). The function then calculates the equilibrium trait value for the response variable after increasing the predictor trait from its median by one median absolute deviation. The function then returns the posterior difference between these values. The resulting $\Delta\theta$ samples can be used to infer whether increases in one trait have a positive or negative selective effect on another trait in the model.

Value

Posterior samples in the `draws_array` format

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRxiv*. doi:10.32942/osf.io/wfp95

Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_calculate_delta_theta](#), [coev_plot_delta_theta](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# calculate delta theta
coev_calculate_delta_theta(
  object = fit,
  response = "political_authority",
  predictor = "religious_authority"
)
```

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

coev_calculate_theta *Calculate equilibrium trait values (theta) for a fitted coevfit object*

Description

Calculate equilibrium trait values θ for one or more traits given a set of intervention values for other traits from a fitted `coevfit` object.

Usage

```
coev_calculate_theta(object, intervention_values = NULL)
```

Arguments

object	An object of class <code>coevfit</code>
intervention_values	Either NULL (the default) or a named list of variables and associated intervention values for calculating equilibrium trait values. If NULL, calculates the equilibrium states when all parameters are free to vary. Otherwise, all coevolving variables must be declared separately in the named list without repetition. If the intervention value for a particular variable is set to NA, this variable is treated as a free variable. If the intervention value for a particular variable is specified, the variable is held constant at this trait value in the calculation. At least one variable must be declared as a free variable and at least one variable must be held constant (e.g., <code>list(var1 = NA, var2 = 0)</code>).

Details

The equilibrium trait values for freely evolving traits η are calculated using the following formula:

$$\theta = -\mathbf{A}^{-1}\mathbf{b}$$

If we hold some variables constant at some value(s) (denoted η_h) and let others evolve freely (denoted η_f), we can partition the parameters as follows:

- $\mathbf{A}_{ff}$: selection coefficients to/from the free variables
- $\mathbf{A}_{fh}$: selection coefficients to the free variables from the held variables
- $\mathbf{b}_f$: continuous time intercepts for the free variables

We can then calculate the equilibrium values for the free variables:

$$\theta_f = -\mathbf{A}_{ff}^{-1} (\mathbf{A}_{fh}\eta_h + \mathbf{b}_f)$$

With the overall equilibrium vector being a mix of the free equilibria and the held values:

$$\theta|\eta_h = \begin{bmatrix} \theta_f \\ \eta_h \end{bmatrix}$$

Value

Posterior samples in matrix format

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

- Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95
- Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_calculate_delta_theta](#), [coev_plot_delta_theta](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# calculate theta with no interventions
coev_calculate_theta(
  object = fit
)

# calculate theta given intervention values
coev_calculate_theta(
  object = fit,
```

```

    intervention_values = list(
      political_authority = NA,
      religious_authority = 0
    )
  )

## End(Not run)

```

coev_fit

Fit Bayesian dynamic coevolutionary model in Stan

Description

Fit Bayesian dynamic coevolutionary models in **Stan** for full Bayesian inference. The model allows users to assess causal directionality ($X \rightarrow Y$ vs. $Y \rightarrow X$) and contingencies (X , then Y) in evolution. Several data types are supported, including binary, ordinal, count, continuous, and positive real variables. The model can additionally account for missing data, repeated observations, and controls for spatial proximity. Model fit can be assessed and compared with posterior predictive checks and cross-validation.

Usage

```

coev_fit(
  data,
  variables,
  id,
  tree,
  effects_mat = NULL,
  complete_cases = FALSE,
  lon_lat = NULL,
  dist_k = NA,
  dist_cov = "exp_quad",
  measurement_error = NULL,
  prior = NULL,
  scale = TRUE,
  estimate_correlated_drift = TRUE,
  estimate_residual = TRUE,
  log_lik = FALSE,
  prior_only = FALSE,
  adapt_delta = 0.95,
  nuts_sampler = "stan",
  backend = deprecated(),
  dist_mat = deprecated(),
  ...
)

```

Arguments

data	An object of class <code>data.frame</code> (or one that can be coerced to that class) containing data of all variables used in the model.
variables	A named list identifying variables that should coevolve in the model and their associated response distributions as character strings (e.g. <code>list(var1 = "bernoulli_logit", var2 = "ordered_logistic")</code>). Variable names must refer to valid column names in data. Currently, the only supported response distributions are <code>bernoulli_logit</code> , <code>ordered_logistic</code> , <code>poisson_softplus</code> , <code>negative_binomial_softplus</code> , <code>normal</code> , and <code>gamma_log</code> . Bernoulli variables must be 0/1 integers, ordered variables must be ordered factors, Poisson and negative binomial variables must be positive integers, normal variables must be continuous numeric, and gamma variables must be positive numeric.
id	A character of length one identifying the variable in the data that links rows to tips on the phylogeny (strictly case-sensitive). Must refer to a valid column name in the data. The id column must exactly match the tip labels in the phylogeny.
tree	A phylogenetic tree object of class <code>phylo</code> or <code>multiPhylo</code> . The tree(s) must be rooted and must include positive non-zero branch lengths. All trees in <code>multiPhylo</code> objects must have the same number of internal nodes and branches.
effects_mat	(optional) A boolean matrix with row and column names exactly matching the variables declared for the model. If not specified, all cross-lagged effects will be estimated in the model. If specified, the model will only estimate cross-lagged effects where cells in the matrix are <code>TRUE</code> and will ignore cross-lagged effects where cells in the matrix are <code>FALSE</code> . In the matrix, columns represent predictor variables and rows represent outcome variables. All autoregressive effects (e.g., <code>X -> X</code>) must be <code>TRUE</code> in the matrix.
complete_cases	(optional) Logical. If <code>FALSE</code> (default), all missing values are imputed by the model. If <code>TRUE</code> , taxa with missing data are excluded.
lon_lat	(optional) A <code>data.frame</code> containing longitude and latitude values for all tips on the phylogeny. The data frame must contain the following columns: a column labelled "id" that exactly matches the tip labels on the phylogeny; a column labelled "longitude" that declares longitude values in decimal degrees; and a column labelled "latitude" that declares latitude values in decimal degrees. If specified, the model will additionally control for spatial location by including a separate Gaussian Process over locations for every coevolving variable in the model.
dist_k	An integer of length one specifying the number of basis functions for computing Hilbert-space approximate Gaussian Processes over locations. If <code>NA</code> (the default), exact Gaussian Processes are computed.
dist_cov	A string of length one specifying the covariance kernel used for Gaussian Processes over locations (strictly case-sensitive). Currently supported are <code>"exp_quad"</code> (exponentiated-quadratic kernel; default), <code>"exponential"</code> (exponential kernel), and <code>"matern32"</code> (Matern 3/2 kernel).
measurement_error	(optional) A named list of coevolving variables and their associated columns in the dataset containing standard errors. Only valid for normally-distributed

variables. For example, if we declare `variables = list(x = "normal", y = "normal")`, then we could set `measurement_error = list(x = "x_std_err")` to tell the function to include measurement error on `x` using standard errors from the `x_std_err` column of the dataset.

prior	(optional) A named list of priors for the model. If not specified, the model uses default priors (see <code>help(coev_fit)</code>). Alternatively, the user can specify a named list of priors. The list must contain non-duplicated entries for any of the following parameters: the autoregressive effects (<code>A_diag</code>), the cross effects (<code>A_offdiag</code>), the Cholesky factor for the drift matrix (<code>L_R</code>), the drift std. dev. parameters (<code>Q_sigma</code>), the continuous time intercepts (<code>b</code>), the ancestral states for the traits (<code>eta_anc</code>), the cutpoints for ordinal variables (<code>c</code>), the overdispersion parameters for negative binomial variables (<code>phi</code>), the shape parameters for gamma variables (<code>shape</code>), the sigma parameters for Gaussian Processes over locations (<code>sigma_dist</code>), the rho parameters for Gaussian Processes over locations (<code>rho_dist</code>), the residual standard deviations when there are repeated observations (<code>sigma_residual</code>), and the Cholesky factor for the residual correlations when there are repeated observations (<code>L_residual</code>). These must be entered with valid prior strings, e.g. <code>list(A_offdiag = "normal(0, 2)")</code> . Invalid prior strings will throw an error when the function internally checks the syntax of resulting Stan code.
scale	Logical. If TRUE (default), variables following the normal and gamma_log response distributions are scaled before fitting the model. Continuous variables following the normal distribution are standardised (e.g., mean centered and divided by their standard deviation) and positive real variables following the gamma_log distribution are divided by the mean value without centering. This approach is recommended when using default priors to improve efficiency and ensure accurate inferences. If FALSE, variables are left unscaled for model fitting. In this case, users should take care to set sensible priors on variables.
estimate_correlated_drift	Logical. If TRUE (default), the model estimates the off-diagonals for the Q drift matrix (i.e., correlated drift). If FALSE, the off-diagonals for the Q drift matrix are set to zero.
estimate_residual	Logical. If TRUE (default), the model estimates residual standard deviations and residual correlations when there are repeated observations for taxa. If FALSE, residual standard deviations and residual correlations are not estimated. The latter may be preferable in cases where repeated observations are sparse (e.g., only some taxa have only few repeated observations). This argument only applies when repeated observations are present in the data.
log_lik	Logical. Set to FALSE by default. If TRUE, the model returns the pointwise log likelihood, which can be used to calculate WAIC and LOO.
prior_only	Logical. If FALSE (default), the model is fitted to the data and returns a posterior distribution. If TRUE, the model samples from the prior only, ignoring the likelihood.
adapt_delta	Target acceptance probability for the NUTS sampler (default 0.95). Passed to <code>cmdstanr::sample()</code> as <code>adapt_delta</code> , or to <code>nutpie</code> as <code>target_accept</code> .

nuts_sampler	Which NUTS implementation to use. "stan" (default) fits the Stan model with CmdStan via cmdstanr . "nutpie" fits a pure JAX model using nutpie's Rust NUTS sampler with JAX gradients via <code>jax.value_and_grad</code> .
backend	Deprecated. Use the nuts_sampler argument instead.
dist_mat	Deprecated. Use the lon_lat argument instead.
...	Additional arguments. For nuts_sampler = "stan": passed to cmdstanr::sample() (and compilation via extra_stanc_args, extra_compile_args). For nuts_sampler = "nutpie": sampling arguments such as chains, iter_sampling, iter_warmup, seed.

Details

The `coev_fit` function generates the **Stan** code for the model, generates the data list, and then compiles and fits the model using the **cmdstanr** package.

Fit a Bayesian dynamic coevolutionary model in Stan. A general overview is provided in the vignette `vignette("coevolve")`

Available response distributions

Variables that should coevolve in the model are declared in the `variables` argument, using a named list with associated response distributions. For example: `list(x = "bernoulli_logit", y = "ordered_logistic")`. Currently, the only supported response distributions are `bernoulli_logit`, `ordered_logistic`, `poisson_softplus`, `negative_binomial_softplus`, `normal`, and `gamma_log`.

Default prior distributions

If priors are not explicitly declared by the user, default priors are used. Default priors were chosen to be weakly regularising, which improves model fitting and conservatism in parameter estimates. Default priors for Stan parameters are as follows:

- `A_diag` (autoregressive effects) = `std_normal()`
- `A_offdiag` (cross effects) = `std_normal()`
- `L_R` (Cholesky factor for drift matrix) = `lkj_corr_cholesky(4)`
- `Q_sigma` (drift std. dev. parameters) = `std_normal()`
- `b` (continuous time intercepts) = `std_normal()`
- `eta_anc` (trait ancestral states) = `std_normal()`
- `c` (ordinal cutpoints) = `normal(0, 2)`
- `shape` (shape parameters for gamma distributions) = `gamma(0.01, 0.01)`
- `sigma_dist` (sigma for Gaussian process over locations) = `exponential(1)`
- `rho_dist` (rho for Gaussian process over locations) = `exponential(5)`
- `sigma_residual` (residual standard deviations) = `exponential(1)`
- `L_residual` (Cholesky factor for residual correlations) = `lkj_corr_cholesky(4)`

The default prior for `phi` (the overdispersion parameter for the negative-binomial distribution) is scaled automatically based on the variance of the data.

We recommend that users assess the suitability of these default priors by fitting the model with `prior_only = TRUE` and then plotting prior predictive checks for all variables using the `coev_plot_predictive_check()` function.

Handling missing data

In order to retain the most information, the `coev_fit()` function automatically imputes all missing values. To turn off this behaviour and exclude taxa with missing data, set `complete_cases = FALSE`.

Dealing with repeated observations

If taxa appear in the dataset multiple times (i.e., there are repeated observations), the model will automatically estimate residual standard deviations and residual correlations that capture the within-taxa variation that is not due to the coevolutionary process. To turn off this behaviour, set `estimate_residual = FALSE`.

We do not recommend modelling residual standard deviations in cases where repeated observations are present for some traits but not others, as there will be no within-species variance for traits without repeated observations.

Note that models with repeated observations for mixtures of continuous and non-continuous traits are currently considered experimental and should be treated with caution.

Incorporating measurement error

If any normally-distributed coevolving variables are measured with error, the user can pass standard errors for one or more of these variables into the model with the `measurement_error` argument.

Controlling for spatial location

If users declare longitude and latitude values with the `lon_lat` argument, the model will include several Gaussian processes (one per coevolving variable) over spatial locations.

Value

Fitted model of class `coevfit`

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

- Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95
- Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_make_stancode](#), [coev_make_standata](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
```

```

    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# print model summary
summary(fit)

## End(Not run)

```

coev_make_stancode

Make Stan code for dynamic coevolutionary model

Description

Make the **Stan** code for the Bayesian dynamic coevolutionary model. **Stan** code is generated, checked for syntactical errors, and then returned as a character string.

Usage

```

coev_make_stancode(
  data,
  variables,
  id,
  tree,
  effects_mat = NULL,
  complete_cases = FALSE,
  lon_lat = NULL,
  dist_k = NA,
  dist_cov = "exp_quad",
  measurement_error = NULL,
  prior = NULL,
  scale = TRUE,
  estimate_correlated_drift = TRUE,
  estimate_residual = TRUE,
  log_lik = FALSE,
  prior_only = FALSE,
  dist_mat = deprecated()
)

```

Arguments

<code>data</code>	An object of class <code>data.frame</code> (or one that can be coerced to that class) containing data of all variables used in the model.
<code>variables</code>	A named list identifying variables that should coevolve in the model and their associated response distributions as character strings (e.g. <code>list(var1 = "bernoulli_logit", var2 = "ordered_logistic")</code>). Variable names must refer to valid column names in <code>data</code> . Currently, the only supported response distributions are <code>bernoulli_logit</code> , <code>ordered_logistic</code> , <code>poisson_softplus</code> , <code>negative_binomial_softplus</code> , <code>normal</code> , and <code>gamma_log</code> . Bernoulli variables must be 0/1 integers, ordered variables must be ordered factors, Poisson and negative binomial variables must be positive integers, normal variables must be continuous numeric, and gamma variables must be positive numeric.
<code>id</code>	A character of length one identifying the variable in the data that links rows to tips on the phylogeny (strictly case-sensitive). Must refer to a valid column name in the data. The <code>id</code> column must exactly match the tip labels in the phylogeny.
<code>tree</code>	A phylogenetic tree object of class <code>phylo</code> or <code>multiPhylo</code> . The tree(s) must be rooted and must include positive non-zero branch lengths. All trees in <code>multiPhylo</code> objects must have the same number of internal nodes and branches.
<code>effects_mat</code>	(optional) A boolean matrix with row and column names exactly matching the variables declared for the model. If not specified, all cross-lagged effects will be estimated in the model. If specified, the model will only estimate cross-lagged effects where cells in the matrix are <code>TRUE</code> and will ignore cross-lagged effects where cells in the matrix are <code>FALSE</code> . In the matrix, columns represent predictor variables and rows represent outcome variables. All autoregressive effects (e.g., <code>X -> X</code>) must be <code>TRUE</code> in the matrix.
<code>complete_cases</code>	(optional) Logical. If <code>FALSE</code> (default), all missing values are imputed by the model. If <code>TRUE</code> , taxa with missing data are excluded.
<code>lon_lat</code>	(optional) A <code>data.frame</code> containing longitude and latitude values for all tips on the phylogeny. The data frame must contain the following columns: a column labelled <code>"id"</code> that exactly matches the tip labels on the phylogeny; a column labelled <code>"longitude"</code> that declares longitude values in decimal degrees; and a column labelled <code>"latitude"</code> that declares latitude values in decimal degrees. If specified, the model will additionally control for spatial location by including a separate Gaussian Process over locations for every coevolving variable in the model.
<code>dist_k</code>	An integer of length one specifying the number of basis functions for computing Hilbert-space approximate Gaussian Processes over locations. If <code>NA</code> (the default), exact Gaussian Processes are computed.
<code>dist_cov</code>	A string of length one specifying the covariance kernel used for Gaussian Processes over locations (strictly case-sensitive). Currently supported are <code>"exp_quad"</code> (exponentiated-quadratic kernel; default), <code>"exponential"</code> (exponential kernel), and <code>"matern32"</code> (Matern 3/2 kernel).
<code>measurement_error</code>	(optional) A named list of coevolving variables and their associated columns in the dataset containing standard errors. Only valid for normally-distributed

	variables. For example, if we declare <code>variables = list(x = "normal", y = "normal")</code> , then we could set <code>measurement_error = list(x = "x_std_err")</code> to tell the function to include measurement error on x using standard errors from the <code>x_std_err</code> column of the dataset.
<code>prior</code>	(optional) A named list of priors for the model. If not specified, the model uses default priors (see <code>help(coev_fit)</code>). Alternatively, the user can specify a named list of priors. The list must contain non-duplicated entries for any of the following parameters: the autoregressive effects (<code>A_diag</code>), the cross effects (<code>A_offdiag</code>), the Cholesky factor for the drift matrix (<code>L_R</code>), the drift std. dev. parameters (<code>Q_sigma</code>), the continuous time intercepts (<code>b</code>), the ancestral states for the traits (<code>eta_anc</code>), the cutpoints for ordinal variables (<code>c</code>), the overdispersion parameters for negative binomial variables (<code>phi</code>), the shape parameters for gamma variables (<code>shape</code>), the sigma parameters for Gaussian Processes over locations (<code>sigma_dist</code>), the rho parameters for Gaussian Processes over locations (<code>rho_dist</code>), the residual standard deviations when there are repeated observations (<code>sigma_residual</code>), and the Cholesky factor for the residual correlations when there are repeated observations (<code>L_residual</code>). These must be entered with valid prior strings, e.g. <code>list(A_offdiag = "normal(0, 2)")</code> . Invalid prior strings will throw an error when the function internally checks the syntax of resulting Stan code.
<code>scale</code>	Logical. If TRUE (default), variables following the normal and gamma_log response distributions are scaled before fitting the model. Continuous variables following the normal distribution are standardised (e.g., mean centered and divided by their standard deviation) and positive real variables following the gamma_log distribution are divided by the mean value without centering. This approach is recommended when using default priors to improve efficiency and ensure accurate inferences. If FALSE, variables are left unscaled for model fitting. In this case, users should take care to set sensible priors on variables.
<code>estimate_correlated_drift</code>	Logical. If TRUE (default), the model estimates the off-diagonals for the Q drift matrix (i.e., correlated drift). If FALSE, the off-diagonals for the Q drift matrix are set to zero.
<code>estimate_residual</code>	Logical. If TRUE (default), the model estimates residual standard deviations and residual correlations when there are repeated observations for taxa. If FALSE, residual standard deviations and residual correlations are not estimated. The latter may be preferable in cases where repeated observations are sparse (e.g., only some taxa have only few repeated observations). This argument only applies when repeated observations are present in the data.
<code>log_lik</code>	Logical. Set to FALSE by default. If TRUE, the model returns the pointwise log likelihood, which can be used to calculate WAIC and LOO.
<code>prior_only</code>	Logical. If FALSE (default), the model is fitted to the data and returns a posterior distribution. If TRUE, the model samples from the prior only, ignoring the likelihood.

dist_mat **[Deprecated]** Use the lon_lat argument instead.

Details

For further details, see `help(coev_fit)`

Value

A character string containing the **Stan** code to fit the dynamic coevolutionary model

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRxiv*. doi:10.32942/osf.io/wfp95

Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_make_standata](#), [coev_fit](#)

Examples

```
# make stan code
stan_code <- coev_make_stancode(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny
)

# include effects matrix
effects_mat <-
  matrix(
    c(TRUE, TRUE,
      FALSE, TRUE),
    nrow = 2,
    dimnames = list(
      c("political_authority", "religious_authority"),
      c("political_authority", "religious_authority")
    )
  )
stan_code <- coev_make_stancode(
```

```

    data = authority$data,
    variables = list(
      political_authority = "ordered_logistic",
      religious_authority = "ordered_logistic"
    ),
    id = "language",
    tree = authority$phylogeny,
    effects_mat = effects_mat
  )

# include longitude/latitude values
stan_code <- coev_make_stancode(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  lon_lat = authority$coordinates
)

# include measurement error
d <- authority$data
d$x <- rnorm(nrow(d))
d$y <- rnorm(nrow(d))
d$x_std_err <- rexp(nrow(d))
d$y_std_err <- rexp(nrow(d))
stan_code <- coev_make_stancode(
  data = d,
  variables = list(
    x = "normal",
    y = "normal"
  ),
  id = "language",
  tree = authority$phylogeny,
  measurement_error = list(
    x = "x_std_err",
    y = "y_std_err"
  )
)

# set manual priors
stan_code <- coev_make_stancode(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  prior = list(A_offdiag = "normal(0, 2)")
)

```

coev_make_standata	<i>Make Stan data for dynamic coevolutionary model</i>
--------------------	--

Description

Make the data list for the **Stan** model. The function takes a dataset, phylogeny, and a set of variables and performs the necessary computations (e.g., cutting up the tree into segments, computing branch lengths, determining parent and child nodes) for the **Stan** model.

Usage

```
coev_make_standata(
  data,
  variables,
  id,
  tree,
  effects_mat = NULL,
  complete_cases = FALSE,
  lon_lat = NULL,
  dist_k = NA,
  dist_cov = "exp_quad",
  measurement_error = NULL,
  prior = NULL,
  scale = TRUE,
  estimate_correlated_drift = TRUE,
  estimate_residual = TRUE,
  log_lik = FALSE,
  prior_only = FALSE,
  dist_mat = deprecated()
)
```

Arguments

data	An object of class <code>data.frame</code> (or one that can be coerced to that class) containing data of all variables used in the model.
variables	A named list identifying variables that should coevolve in the model and their associated response distributions as character strings (e.g. <code>list(var1 = "bernoulli_logit", var2 = "ordered_logistic")</code>). Variable names must refer to valid column names in <code>data</code> . Currently, the only supported response distributions are <code>bernoulli_logit</code> , <code>ordered_logistic</code> , <code>poisson_softplus</code> , <code>negative_binomial_softplus</code> , <code>normal</code> , and <code>gamma_log</code> . Bernoulli variables must be 0/1 integers, ordered variables must be ordered factors, Poisson and negative binomial variables must be positive integers, normal variables must be continuous numeric, and gamma variables must be positive numeric.

id	A character of length one identifying the variable in the data that links rows to tips on the phylogeny (strictly case-sensitive). Must refer to a valid column name in the data. The id column must exactly match the tip labels in the phylogeny.
tree	A phylogenetic tree object of class <code>phylo</code> or <code>multiPhylo</code> . The tree(s) must be rooted and must include positive non-zero branch lengths. All trees in <code>multiPhylo</code> objects must have the same number of internal nodes and branches.
effects_mat	(optional) A boolean matrix with row and column names exactly matching the variables declared for the model. If not specified, all cross-lagged effects will be estimated in the model. If specified, the model will only estimate cross-lagged effects where cells in the matrix are <code>TRUE</code> and will ignore cross-lagged effects where cells in the matrix are <code>FALSE</code> . In the matrix, columns represent predictor variables and rows represent outcome variables. All autoregressive effects (e.g., $X \rightarrow X$) must be <code>TRUE</code> in the matrix.
complete_cases	(optional) Logical. If <code>FALSE</code> (default), all missing values are imputed by the model. If <code>TRUE</code> , taxa with missing data are excluded.
lon_lat	(optional) A data frame containing longitude and latitude values for all tips on the phylogeny. The data frame must contain the following columns: a column labelled "id" that exactly matches the tip labels on the phylogeny; a column labelled "longitude" that declares longitude values in decimal degrees; and a column labelled "latitude" that declares latitude values in decimal degrees. If specified, the model will additionally control for spatial location by including a separate Gaussian Process over locations for every coevolving variable in the model.
dist_k	An integer of length one specifying the number of basis functions for computing Hilbert-space approximate Gaussian Processes over locations. If <code>NA</code> (the default), exact Gaussian Processes are computed.
dist_cov	A string of length one specifying the covariance kernel used for Gaussian Processes over locations (case-sensitive). Currently supported are "exp_quad" (exponentiated-quadratic kernel; default), "exponential" (exponential kernel), and "matern32" (Matern 3/2 kernel).
measurement_error	(optional) A named list of coevolving variables and their associated columns in the dataset containing standard errors. Only valid for normally-distributed variables. For example, if we declare <code>variables = list(x = "normal", y = "normal")</code> , then we could set <code>measurement_error = list(x = "x_std_err")</code> to tell the function to include measurement error on x using standard errors from the <code>x_std_err</code> column of the dataset.
prior	(optional) A named list of priors for the model. If not specified, the model uses default priors (see <code>help(coev_fit)</code>). Alternatively, the user can specify a named list of priors. The list must contain non-duplicated entries for any of the following parameters: the autoregressive effects (<code>A_diag</code>), the cross effects (<code>A_offdiag</code>), the Cholesky factor for the drift matrix (<code>L_R</code>), the drift std. dev. parameters (<code>Q_sigma</code>), the continuous time intercepts (<code>b</code>), the ancestral states for the traits (<code>eta_anc</code>), the cutpoints for ordinal variables (<code>c</code>), the overdispersion parameters for negative binomial variables (<code>phi</code>), the shape parameters for gamma variables (<code>shape</code>), the sigma parameters for Gaussian Processes over locations (<code>sigma_dist</code>), the rho parameters for Gaussian Processes over locations

	(rho_dist), the residual standard deviations when there are repeated observations (sigma_residual), and the Cholesky factor for the residual correlations when there are repeated observations (L_residual). These must be entered with valid prior strings, e.g. <code>list(A_offdiag = "normal(0, 2)")</code> .
scale	Logical. If TRUE (default), variables following the normal and gamma_log response distributions are scaled before fitting the model. Continuous variables following the normal distribution are standardised (e.g., mean centered and divided by their standard deviation) and positive real variables following the gamma_log distribution are divided by the mean value without centering. This approach is recommended when using default priors to improve efficiency and ensure accurate inferences. If FALSE, variables are left unscaled for model fitting. In this case, users should take care to set sensible priors on variables.
estimate_correlated_drift	Logical. If TRUE (default), the model estimates the off-diagonals for the Q drift matrix (i.e., correlated drift). If FALSE, the off-diagonals for the Q drift matrix are set to zero.
estimate_residual	Logical. If TRUE (default), the model estimates residual standard deviations and residual correlations when there are repeated observations for taxa. If FALSE, residual standard deviations and residual correlations are not estimated. The latter may be preferable in cases where repeated observations are sparse (e.g., only some taxa have only few repeated observations). This argument only applies when repeated observations are present in the data.
log_lik	Logical. Set to FALSE by default. If TRUE, the model returns the pointwise log likelihood, which can be used to calculate WAIC and LOO.
prior_only	Logical. If FALSE (default), the model is fitted to the data and returns a posterior distribution. If TRUE, the model samples from the prior only, ignoring the likelihood.
dist_mat	[Deprecated] Use the <code>lon_lat</code> argument instead.

Details

For further details, see `help(coev_fit)`

Value

A list containing the data for fitting the dynamic coevolutionary model in **Stan**

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

- Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRxiv*. doi:10.32942/osf.io/wfp95
- Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_make_stancode](#), [coev_fit](#)

Examples

```
# make stan data
stan_data <- coev_make_standata(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny
)

# include effects matrix
effects_mat <-
  matrix(
    c(TRUE, TRUE,
      FALSE, TRUE),
    nrow = 2,
    dimnames = list(
      c("political_authority", "religious_authority"),
      c("political_authority", "religious_authority")
    )
  )

stan_data <- coev_make_standata(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  effects_mat = effects_mat
)

# include longitude/latitude values
stan_data <- coev_make_standata(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
```

```

      religious_authority = "ordered_logistic"
    ),
    id = "language",
    tree = authority$phylogeny,
    lon_lat = authority$coordinates
  )

# include measurement error
d <- authority$data
d$x <- rnorm(nrow(d))
d$y <- rnorm(nrow(d))
d$x_std_err <- rexp(nrow(d))
d$y_std_err <- rexp(nrow(d))
stan_data <- coev_make_standata(
  data = d,
  variables = list(
    x = "normal",
    y = "normal"
  ),
  id = "language",
  tree = authority$phylogeny,
  measurement_error = list(
    x = "x_std_err",
    y = "y_std_err"
  )
)

# set manual priors
stan_data <- coev_make_standata(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  prior = list(A_offdiag = "normal(0, 2)")
)

```

coev_plot_delta_theta *Plot delta theta values from a fitted coevfit object*

Description

Plot delta theta values for all trait pairs from a fitted coevfit object. This plot can be used to visually assess contingencies and directionality between different variables in the coevolutionary process.

Usage

```

coev_plot_delta_theta(
  object,
  variables = NULL,
  prob = 0.66,
  prob_outer = 0.95,
  limits = NULL
)

```

Arguments

object	An object of class <code>coevfit</code>
variables	If <code>NULL</code> (default), the plot includes all coevolving variables from the model. Otherwise, a character vector of length ≥ 2 declaring the variables to be included in the plot.
prob	Probability mass to include in the inner interval. Default is 0.66 (66% interval).
prob_outer	Probability mass to include in the outer interval. Default is 0.95 (95% interval).
limits	If <code>NULL</code> (default), limits are scaled automatically to include all posterior samples. Otherwise, a numeric vector of length 2 specifying the lower and upper limits for the x-axis.

Details

This function repeatedly uses the `coev_calculate_delta_theta` function under the hood to generate a pairs plot of $\Delta\theta$ for all variables in the model. For more details on the definition and calculation of $\Delta\theta$, see `help(coev_calculate_delta_theta)`. Note that often the posterior distribution for $\Delta\theta$ has long tails, meaning that the distribution for different traits can be difficult to visualise in a single pairs plot. If this plot does not produce satisfactory visualisations, the user should either specify narrower limits for the x-axis or use the `coev_calculate_delta_theta` function to create plots manually.

Value

A ggplot object

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

- Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95
- Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_calculate_delta_theta](#), [coev_calculate_theta](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# plot delta theta values for all effects
coev_plot_delta_theta(fit)

## End(Not run)
```

coev_plot_flowfield	<i>Plot flowfield of expected evolutionary change from a fitted coevfit object</i>
---------------------	--

Description

Plot a flowfield of the expected evolutionary dynamics from a fitted `coevfit` object, with help from the **phaseR** package.

Usage

```
coev_plot_flowfield(
  object,
  var1,
  var2,
  nullclines = FALSE,
  limits = c(-2.5, 2.5)
)
```

Arguments

object	An object of class <code>coevfit</code>
var1	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)
var2	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)
nullclines	Logical (defaults to FALSE); whether to show coloured nullclines to indicate where each variable is at equilibrium, depending on the state of the other
limits	A numeric vector of length 2 (defaults to <code>c(-2.5, 2.5)</code>); specifying the lower limit and the upper limit of the x and y axes.

Details

The flowfield visualises the change in two coevolving traits depending on their current states, with the direction and strength of change depicting with the direction and size of arrows. If nullclines are included, they represent the parameter combinations where each trait is at equilibrium, depending on the state of the other trait.

If three or more traits were included in the model, other traits are held at their median values during the computations. Note that flowfield plots can potentially produce misleading pictures of coevolutionary dynamics when other traits are held constant in models with three or more traits.

If the plot does not look right, the user might try zooming out from the default parameter space by setting wider limits. For some variables (e.g., continuous and count variables), the default limits may not be suitable.

Value

A flowfield plot drawn directly to the device

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

- Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95
- Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_plot_selection_gradient](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# plot flow field
coev_plot_flowfield(
  object = fit,
  var1 = "political_authority",
  var2 = "religious_authority"
)

## End(Not run)
```

`coev_plot_pred_series` *Plot a predicted coevolutionary time series from a fitted coevfit object*

Description

This function plots a predicted coevolutionary time series using the estimated parameters from a fitted `coevfit` model. By default, the plot uses the posterior ancestral states estimated by the model as the starting values, but users can also set their own starting values for traits. Plots can be generated with or without stochastic drift. For more details on the underlying predictive function, see `help(coev_pred_series)`.

Usage

```
coev_plot_pred_series(object, prob = 0.95, ...)
```

Arguments

<code>object</code>	An object of class <code>coevfit</code>
<code>prob</code>	A value between 0 and 1 indicating the desired probability to be covered by the uncertainty intervals. The default is 0.95.
<code>...</code>	Additional arguments passed to coev_pred_series

Value

A ggplot object

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

See Also

[coev_pred_series](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# simulated trait co-evolution
coev_plot_pred_series(
  object = fit,
  stochastic = TRUE
)

# expected trait co-evolution, no drift
coev_plot_pred_series(
  object = fit,
  stochastic = FALSE,
  eta_anc = list(political_authority = -2, religious_authority = 1.5)
)

## End(Not run)
```

coev_plot_predictive_check

Plot predictive check from a fitted coevfit object

Description

Plot posterior (or prior) predictive checks for variables from a fitted `coevfit` object, with help from the **bayesplot** package.

Usage

```
coev_plot_predictive_check(
  object,
  variables = NULL,
  ndraws = NULL,
  tree_id = NULL
)
```

Arguments

<code>object</code>	An object of class <code>coevfit</code>
<code>variables</code>	If <code>NULL</code> (default), the function returns a list of plots for all coevolving variables from the model. Otherwise, a character vector declaring the variables to be included in the list of plots.
<code>ndraws</code>	An integer indicating the number of draws to return. The default and maximum number of draws is the size of the posterior sample.
<code>tree_id</code>	An integer indicating the tree ID to use when making posterior predictions. Set to <code>NULL</code> by default, which will use draws from every tree, integrating phylogenetic uncertainty.

Value

A list of ggplot objects

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)
```

```

)

# plot predictive checks for all variables
coev_plot_predictive_check(fit)

## End(Not run)

```

coev_plot_selection_gradient

Plot selection gradient heatmap from a fitted coevfit object

Description

Plot a heatmap of the selection gradient for two variables from a fitted coevfit object.

Usage

```

coev_plot_selection_gradient(
  object,
  var1,
  var2,
  contour = FALSE,
  limits = c(-2.5, 2.5)
)

```

Arguments

object	An object of class coevfit
var1	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)
var2	A character string of length one equal to one of the coevolving variables in the model (strictly case-sensitive)
contour	Logical (defaults to FALSE); whether to show white contour lines to indicate where selection is stronger than drift
limits	A numeric vector of length 2 (defaults to c(-2.5, 2.5)); specifying the lower limit and the upper limit of the x and y axes.

Details

The selection gradient is operationalised as the ratio of the change in a trait due to deterministic selection $\Delta\alpha$ to the change in a trait due to stochastic drift σ . Values between -1 and 1 indicate parameter space where the change due to drift is greater than change due to selection on the trait. Conversely, values greater than 1 (or less than -1) indicate parameter space where positive (or negative) selection is stronger than drift.

If three or more traits were included in the model, other traits are held at their median values during the computations. Note that selection gradient plots can potentially produce misleading pictures of coevolutionary dynamics when other traits are held constant in models with three or more traits.

If the plot does not look right, the user might try zooming out from the default parameter space by setting wider limits. For some variables (e.g., continuous and count variables), the default limits may not be suitable.

Value

A ggplot object

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95

Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

See Also

[coev_plot_flowfield](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# plot selection gradient
coev_plot_selection_gradient(
  object = fit,
  var1 = "political_authority",
  var2 = "religious_authority"
)
```

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

```
coev_plot_trait_values
```

Plot estimated trait values from a fitted coevfit object

Description

Produce a pairs plot of the estimated trait values for taxa from a fitted `coevfit` object. The plot includes scatterplot(s) of median estimated trait values, heatmap(s) of median estimated trait values, and density plots of the marginal distributions for each variable with associated posterior uncertainty.

Usage

```
coev_plot_trait_values(
  object,
  variables = NULL,
  ndraws = 50,
  tree_id = NULL,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

<code>object</code>	An object of class <code>coevfit</code>
<code>variables</code>	If <code>NULL</code> (default), the function returns a pairs plot including all coevolving variables from the model. Otherwise, a character vector declaring the variables to be included.
<code>ndraws</code>	An integer indicating the number of draws to return in the density plots on the diagonal. The default and maximum number of draws is the size of the posterior sample.
<code>tree_id</code>	An integer indicating the tree ID to use when making posterior predictions. Set to <code>NULL</code> by default, which will use draws from every tree, integrating phylogenetic uncertainty.
<code>xlim</code>	Limits for the x-axis. If <code>NULL</code> (default), limits are set to the minimum and maximum estimated trait values.
<code>ylim</code>	Limits for the y-axis. If <code>NULL</code> (default), limits are set to the minimum and maximum estimated trait values.

Value

A patchwork of ggplot objects

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

References

Ringen, E., Martin, J. S., & Jaeggi, A. (2021). Novel phylogenetic methods reveal that resource-use intensification drives the evolution of "complex" societies. *EcoEvoRXiv*. doi:10.32942/osf.io/wfp95

Sheehan, O., Watts, J., Gray, R. D., Bulbulia, J., Claessens, S., Ringen, E. J., & Atkinson, Q. D. (2023). Coevolution of religious and political authority in Austronesian societies. *Nature Human Behaviour*, 7(1), 38-45. 10.1038/s41562-022-01471-y

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# pairs plot of trait values
coev_plot_trait_values(fit)

## End(Not run)
```

coev_pred_series

Predict a co-evolutionary time series from a fitted coevfit object

Description

This function produces predicted values of traits over a time series, given the estimated parameters for a coevfit model. This function is used under the hood by the plotting function [coev_pred_series](#).

Usage

```
coev_pred_series(
  object,
  eta_anc = NULL,
```

```

intervention_values = NULL,
tmax = 1,
ntimes = 30,
ndraws = NULL,
stochastic = FALSE
)

```

Arguments

object	An object of class <code>coevfit</code>
eta_anc	If NULL (default), the starting values for the latent states η will be set to the estimated ancestral states at the root from the <code>coevfit</code> model. Otherwise, a named list with length equal to the number of variables specifying the initial η values. All variable names must be included in this list.
intervention_values	Either NULL (the default) or a named list of variables and associated intervention values. If NULL, all traits are free to vary. Otherwise, all coevolving variables must be declared separately in the named list without repetition. If the intervention value for a particular variable is set to NA, this variable is treated as a free variable. If the intervention value for a particular variable is specified, the variable is held constant at this trait value in the calculation. At least one variable must be declared as a free variable and at least one variable must be held constant (e.g., <code>list(var1 = NA, var2 = 0)</code>).
tmax	A positive number indicating the total duration of time for which to predict. Set to 1 by default, corresponding to the entire time depth of the original phylogenetic tree(s).
ntimes	A positive integer indicating the total number of discrete time steps over which to make predictions. Each step corresponds to a time difference of $dt = tmax/ntimes$. Set to 30 by default.
ndraws	An integer indicating the number of draws to return. The default and maximum number of draws is the size of the posterior sample.
stochastic	Logical (defaults to FALSE); indicator of whether predictions should include only the expected co-evolutionary change due to deterministic selection (FALSE) or also stochastic drift (TRUE). Incompatible with the 'intervention_values' argument.

Value

An [ndraw, ntimes, nvariables] array of predicted η values

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

See Also

[coev_plot_pred_series](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# simulate trait co-evolution
sims <- coev_pred_series(
  object = fit,
  stochastic = TRUE
)

# expected trait co-evolution, no drift
epreds <- coev_pred_series(
  object = fit,
  stochastic = FALSE,
  eta_anc = list(political_authority = -2, religious_authority = 1.5)
)

# expected trait co-evolution under intervention, no drift
epreds_intervention <- coev_pred_series(
  object = fit,
  stochastic = FALSE,
  intervention_values = list(political_authority = NA,
                             religious_authority = 1.5)
)

## End(Not run)
```

coev_simulate_coevolution

Simulate the coevolution of multiple variables in discrete time steps

Description

This function simulates the coevolution of multiple continuous variables in discrete time steps following a simple VAR(1) autoregressive model. Users set the sample size, the variable names, the

strength of selection and drift, and the probability of a speciation event in a given time step. The function returns a phylogeny, the results of the simulation run, and a dataset of contemporary trait values.

Usage

```
coev_simulate_coevolution(
  n,
  variables,
  selection_matrix,
  drift,
  prob_split,
  intercepts = NULL,
  ancestral_states = NULL
)
```

Arguments

<code>n</code>	Number of data points in the resulting data frame.
<code>variables</code>	A character vector of variable names (e.g., <code>c("x", "y")</code>)
<code>selection_matrix</code>	A numeric matrix determining the strength of selection between variables. The matrix must have a number of rows and columns equal to the number of variables and its row and column names must contain all specified variables. Each cell determines the strength of selection from the column variable to the row variable. For example, the cell on the "x" column and the "x" row indicates how much previous values of x influence future values of x (autocorrelation). By contrast, the cell on the "x" column and the "y" row indicates how much previous values of x influence future values of y (cross-lagged effect).
<code>drift</code>	A named numeric vector specifying the strength of drift for different variables. Names must include all specified variables.
<code>prob_split</code>	A numeric probability of a species split in any given timestep.
<code>intercepts</code>	Intercepts for the VAR(1) model. If NULL (default), intercepts are set to zero for all variables. Otherwise, a named numeric vector specifying the intercepts for different variables. Names must include all specified variables.
<code>ancestral_states</code>	Ancestral states for different variables. If NULL (default), ancestral states are set to zero for all variables. Otherwise, a named numeric vector specifying the ancestral states for different variables. Names must include all specified variables.

Details

The model underlying this simulation is a simple VAR(1) autoregressive model, where values of all variables at the previous timestep predict values at the current timestep. In the case of two variables, the model is as follows:

$$Y_t = \alpha_y + \beta_{y,y}Y_{t-1} + \beta_{y,x}X_{t-1} + \mathcal{N}(0, \epsilon_y)$$

$$X_t = \alpha_x + \beta_{x,x}X_{t-1} + \beta_{x,y}Y_{t-1} + \mathcal{N}(0, \epsilon_x)$$

where α represents the intercepts, β represents the selection matrix, and ϵ represents the vector of drift parameters. With some probability p , a speciation event creates two independent evolutionary branches. This simulation continues until the intended sample size of species has been reached. Note: this VAR(1) model is not the same as the generalized dynamic phylogenetic model that is estimated by the `coev_fit()` function (see the introductory vignette for mathematical details of generalized dynamic phylogenetic models).

Value

List with dataset at final timestep (data), full simulation log (simulation), and pruned phylogenetic tree (tree)

Author(s)

Scott Claessens <scott.claessens@gmail.com>, Erik Ringen <erikjacob.ringen@uzh.ch>

Examples

```
# simulate coevolution of x and y
n <- 100
variables <- c("x", "y")
# x -> y but not vice versa
selection_matrix <- matrix(
  c(
    0.95, 0.00,
    0.80, 0.95
  ),
  nrow = 2,
  byrow = TRUE,
  dimnames = list(variables, variables)
)
drift <- c("x" = 0.05, "y" = 0.05)
prob_split <- 0.05

# run simulation with default parameters
coev_simulate_coevolution(
  n, variables, selection_matrix,
  drift, prob_split
)

# set intercepts manually
coev_simulate_coevolution(
  n, variables, selection_matrix,
  drift, prob_split,
  intercepts = c("x" = 1, "y" = -1)
)

# set ancestral states manually
coev_simulate_coevolution(
  n, variables, selection_matrix,
```

```

    drift, prob_split,
    ancestral_states = c("x" = 1, "y" = -1)
  )

```

draws.jax_fit	<i>Extract draws from jax_fit object</i>
---------------	--

Description

Internal S3 method. Extracts draws from a jax_fit object, compatible with cmdstanr's draws() method.

Usage

```

## S3 method for class 'jax_fit'
draws(x, variables = NULL, ...)

```

Arguments

x	A jax_fit object.
variables	Character vector of variable names to extract.
...	Additional arguments (ignored).

Value

A draws_array object.

Examples

```
TRUE
```

draws.nutpie_fit	<i>Extract draws from nutpie_fit object</i>
------------------	---

Description

Internal S3 method. Extracts draws from a nutpie_fit object, compatible with cmdstanr's draws() method.

Usage

```

## S3 method for class 'nutpie_fit'
draws(x, variables = NULL, ...)

```

Arguments

`x` A `nutpie_fit` object.

`variables` Character vector of variable names to extract. If `NULL`, extracts all variables.

`...` Additional arguments (currently ignored).

Value

A `draws_array` object.

Examples

```
TRUE
```

<code>extract_samples</code>	<i>Extract samples (draws) from a fitted <code>coevfit</code> object</i>
------------------------------	--

Description

Extract samples (draws) from a fitted `coevfit` object

Usage

```
extract_samples(object)
```

Arguments

`object` An object of class `coevfit`

Value

Samples in 'rethinking' style list format

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
```

```

      seed = 1
    )

    # extract samples from fitted model
    extract_samples(fit)

    ## End(Not run)

```

metadata.jax_fit	<i>Extract metadata from jax_fit object</i>
------------------	---

Description

Internal S3 method. Extracts metadata from a jax_fit object, compatible with cmdstanr's metadata() method.

Usage

```

## S3 method for class 'jax_fit'
metadata(x)

```

Arguments

x A jax_fit object.

Value

A list containing metadata.

Examples

```
TRUE
```

metadata.nutpie_fit	<i>Extract metadata from nutpie_fit object</i>
---------------------	--

Description

Internal S3 method. Extracts metadata from a nutpie_fit object, compatible with cmdstanr's metadata() method.

Usage

```

## S3 method for class 'nutpie_fit'
metadata(x)

```

Arguments

x A nutpie_fit object.

Value

A list containing metadata.

Examples

```
TRUE
```

pairs.coevfit	<i>Create a matrix of output plots from a coevfit object</i>
---------------	--

Description

A [pairs](#) method that is customized for MCMC output.

Usage

```
## S3 method for class 'coevfit'
pairs(x, parameters = NULL, ...)
```

Arguments

x	An object of class coevfit
parameters	If NULL (default), the function returns plots for the main parameters from the model (the selection matrix "A", the drift matrix "Q", and the continuous time intercepts "b"). Otherwise, a character vector declaring the parameters to plot (strictly case-sensitive).
...	Further arguments to be passed to mcmc_pairs .

Details

For a detailed description see [mcmc_pairs](#).

Value

A matrix of output plots

Author(s)

Scott Claessens <scott.claessens@gmail.com>

See Also

[plot.coevfit](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# print pairs plot
pairs(fit)

## End(Not run)
```

plot.coevfit

Trace and density plots for coevfit objects

Description

Trace and density plots for coevfit objects

Usage

```
## S3 method for class 'coevfit'
plot(
  x,
  parameters = NULL,
  combo = c("dens", "trace"),
  npars = 5,
  plot = TRUE,
  ...
)
```

Arguments

x	An object of class coevfit.
parameters	If NULL (default), the function returns plots for the main parameters from the model (the selection matrix "A", the drift matrix "Q", and the continuous time intercepts "b"). Otherwise, a character vector declaring the parameters to plot (strictly case-sensitive).

combo	A character vector with at least two elements. Each element of combo corresponds to a column in the resulting plot and should be the name of one of the available MCMC functions (omitting the <code>mcmc_</code> prefix).
npars	Number of parameters displayed per page (default = 5).
plot	Logical. If TRUE (default), plots are plotted in the active graphic device. If FALSE, plots are not plotted.
...	Not used.

Value

A list of ggplot objects

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# print trace and density plots
plot(fit)

## End(Not run)
```

primates

Brain weight, body weight, diet, and sociality in primates

Description

This dataset and associated phylogeny describes brain weights, body weights, dietary categories, social systems, mating systems, and group sizes for 143 primate species. These data were originally compiled in DeCasien et al. (2017). The phylogeny is a 50% majority rule consensus tree from the 10k Trees website.

Usage

```
primates
```

Format

A list containing a dataset and an associated phylogeny. The phylogeny is a pruned consensus tree for 143 species of primate (excluding humans). The dataset is a data frame with 143 observations and 11 variables:

species The primate species

clade The clade of the primate species

brain_weight Average brain weight, in grams

body_weight Average body weight, in grams

diet A factor with four levels indicating whether the species' dietary category is folivore, frugivore, frugivore/folivore, or omnivore

percent_fruit Percentage of fruit in diet

social_system A factor with four levels indicating whether the species' social system is solitary, pair-living, harem polygyny, or polygynandry

mating_system A factor with five levels indicating whether the species' mating system is spatial polygyny, monogamy, polyandry, harem polygyny, or polygynandry

group_size Average group size

Source

DeCasien, A. R., Williams, S. A., & Higham, J. P. (2017). Primate brain size is predicted by diet but not sociality. *Nature Ecology & Evolution*, 1(5), 0112.

Arnold, C., Matthews, L. J., & Nunn, C. L. (2010). The 10kTrees website: a new online resource for primate phylogeny. *Evolutionary Anthropology*, 19(3), 114-118.

<https://10ktrees.nunn-lab.org/Primates/index.html>

Examples

```
## Not run:
# fit model to primates data
m <-
  coev_fit(
    data = primates$data,
    variables = list(
      brain_weight = "lognormal",
      body_weight = "lognormal"
    ),
    id = "species",
    tree = primates$phylogeny,
    # arguments to cmdstanr::sample()
    parallel_chains = 4,
    seed = 1
  )
# print model summary
summary(m)
# plot delta theta
coev_plot_delta_theta(m)
```

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

print.coevfit	<i>Print a summary of a fitted model represented by a coevfit object</i>
---------------	--

Description

Print a summary of a fitted model represented by a coevfit object

Usage

```
## S3 method for class 'coevfit'
print(x, digits = 2, ...)
```

Arguments

x	An object of class coevfit
digits	The number of significant digits for printing out the summary; defaults to 2
...	Additional arguments that would be passed to method summary of coevfit.

Value

A printed summary of a coevfit model object

See Also

[summary.coevfit](#)

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# print summary of model
```

```
print(fit)

## End(Not run)
```

repeated

Example dataset with repeated observations

Description

This example dataset and associated phylogeny are used as an example of repeated observations in phylogenetic modelling. The data are adapted from de Villemereuil & Nakagawa (2014) to include only the first twenty species.

Usage

```
repeated
```

Format

A list containing a dataset and an associated phylogeny. The dataset is a data frame with 100 observations and 3 variables:

species The name of the species
x An example continuous variable
y An example continuous variable

Source

de Villemereuil P. & Nakagawa, S. (2014). General quantitative genetic methods for comparative biology. In L. Garamszegi (Ed.), *Modern phylogenetic comparative methods and their application in evolutionary biology: concepts and practice* (pp. 287-303). Springer, New York.

Examples

```
## Not run:
# fit model to repeated data
m <-
  coev_fit(
    data = repeated$data,
    variables = list(
      x = "normal",
      y = "normal"
    ),
    id = "species",
    tree = repeated$phylogeny,
    # arguments to cmdstanr::sample()
    parallel_chains = 4,
    seed = 1
```

```

    )
  # print model summary
  summary(m)
  # plot delta theta
  coev_plot_delta_theta(m)

  ## End(Not run)

```

save_coevfit

Save a fitted coevfit object to a file

Description

A wrapper around [saveRDS](#) that saves a fitted coevfit model object while ensuring that all posterior draws and diagnostics are correctly saved by **cmdstanr**.

Usage

```
save_coevfit(object, file, ...)
```

Arguments

object	An object of class coevfit
file	A string declaring the path where the file should be saved
...	Other arguments to pass to saveRDS besides object and file

Value

An .RDS file containing the fitted coevfit model object

Examples

```

## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

```

```
# save model to .rds file
save_coevfit(fit, file = "model.rds")

## End(Not run)
```

stancode	<i>Expose the Stan code of a fitted model represented by a coevfit object</i>
----------	---

Description

Expose the Stan code of a fitted model represented by a coevfit object

Usage

```
stancode(object)
```

Arguments

object An object of class coevfit.

Value

Printed Stan code

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# expose stan code
stancode(fit)

## End(Not run)
```

standata	<i>Expose the Stan data list for a fitted model represented by a coevfit object</i>
----------	---

Description

Expose the Stan data list for a fitted model represented by a coevfit object

Usage

```
standata(object)
```

Arguments

object An object of class coevfit.

Value

Named list of data for Stan

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
  tree = authority$phylogeny,
  # additional arguments for cmdstanr::sample()
  chains = 4,
  parallel_chains = 4,
  seed = 1
)

# expose stan data
standata(fit)

## End(Not run)
```

summary.coevfit	Create a summary of a fitted model represented by a coevfit object
-----------------	--

Description

Create a summary of a fitted model represented by a coevfit object

Usage

```
## S3 method for class 'coevfit'
summary(object, prob = 0.95, robust = FALSE, ...)
```

Arguments

object	An object of class coevfit.
prob	A value between 0 and 1 indicating the desired probability to be covered by the uncertainty intervals. The default is 0.95.
robust	If FALSE (the default) the mean is used as the measure of central tendency and the standard deviation as the measure of variability. If TRUE, the median and the median absolute deviation are applied instead.
...	Other potential arguments

Details

The convergence diagnostics rhat, ess_bulk, and ess_tail are described in detail in Vehtari et al. (2020).

Value

List of class coevsummary

References

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner (2020). Rank-normalization, folding, and localization: An improved R-hat for assessing convergence of MCMC. *Bayesian Analysis*. 1–28. doi:10.1214/20-BA1221

Examples

```
## Not run:
# fit dynamic coevolutionary model
fit <- coev_fit(
  data = authority$data,
  variables = list(
    political_authority = "ordered_logistic",
    religious_authority = "ordered_logistic"
  ),
  id = "language",
```

```

tree = authority$phylogeny,
# additional arguments for cmdstanr::sample()
chains = 4,
parallel_chains = 4,
seed = 1
)

# print summary of model
summary(fit)

## End(Not run)

```

summary.jax_fit

Summary statistics for jax_fit object

Description

Internal S3 method. Computes summary statistics from a `jax_fit` object, compatible with `cmdstanr`'s `summary()` method. This method mimics `cmdstanr`'s interface where `summary()` is called with named arguments where names are column names and values are function names or formulas.

Usage

```

## S3 method for class 'jax_fit'
summary(object, variables = NULL, ...)

```

Arguments

<code>object</code>	A <code>jax_fit</code> object.
<code>variables</code>	Character vector of variable names to summarize.
<code>...</code>	Named summary spec arguments (<code>cmdstanr</code> style).

Value

A `data.frame` with summary statistics.

Examples

```
TRUE
```

summary.nutpie_fit	<i>Compute summary statistics for nutpie_fit object</i>
--------------------	---

Description

Internal S3 method. Computes summary statistics from a nutpie_fit object, compatible with cmdstanr's summary() method. This method mimics cmdstanr's interface where summary() is called with named arguments where names are column names and values are function names or formulas.

Usage

```
## S3 method for class 'nutpie_fit'  
summary(object, variables = NULL, ...)
```

Arguments

object	A nutpie_fit object.
variables	Character vector of variable names to summarize. If NULL, summarizes all variables.
...	Named arguments where names are output column names and values are function names (strings) or formulas for computing statistics.

Value

A data.frame with summary statistics, compatible with cmdstanr format.

Examples

```
TRUE
```

Index

* datasets

- authority, [3](#)
- primates, [44](#)
- repeated, [47](#)

authority, [3](#)

coev_ancestral_states, [4](#)

coev_calculate_delta_theta, [6](#), [7](#), [9](#), [25](#),
[26](#)

coev_calculate_theta, [7](#), [8](#), [26](#)

coev_fit, [5](#), [10](#), [18](#), [23](#)

coev_make_stancode, [14](#), [15](#), [23](#)

coev_make_standata, [14](#), [18](#), [20](#)

coev_plot_delta_theta, [7](#), [9](#), [24](#)

coev_plot_flowfield, [26](#), [32](#)

coev_plot_pred_series, [28](#), [35](#)

coev_plot_predictive_check, [29](#)

coev_plot_selection_gradient, [27](#), [31](#)

coev_plot_trait_values, [5](#), [33](#)

coev_pred_series, [28](#), [29](#), [34](#), [34](#)

coev_simulate_coevolution, [36](#)

draws.jax_fit, [39](#)

draws.nutpie_fit, [39](#)

extract_samples, [40](#)

MCMC, [44](#)

mcmc_pairs, [42](#)

metadata.jax_fit, [41](#)

metadata.nutpie_fit, [41](#)

pairs, [42](#)

pairs.coevfit, [42](#)

plot.coevfit, [42](#), [43](#)

primates, [44](#)

print.coevfit, [46](#)

print.coevsummary (print.coevfit), [46](#)

repeated, [47](#)

save_coevfit, [48](#)

saveRDS, [48](#)

stancode, [49](#)

standata, [50](#)

summary.coevfit, [46](#), [51](#)

summary.jax_fit, [52](#)

summary.nutpie_fit, [53](#)