Package: phyr (via r-universe)

October 23, 2024

Type Package

Title Model Based Phylogenetic Analysis

Version 1.1.0

Description A collection of functions to do model-based phylogenetic analysis. It includes functions to calculate community phylogenetic diversity, to estimate correlations among functional traits while accounting for phylogenetic relationships, and to fit phylogenetic generalized linear mixed models. The Bayesian phylogenetic generalized linear mixed models are fitted with the 'INLA' package (<https://www.r-inla.org>).

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.1)

Imports stats, ape, Rcpp, Matrix, methods, graphics, dplyr, lme4, nloptr, gridExtra, mvtnorm, latticeExtra, tidyr

RoxygenNote 7.1.1

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, pez, knitr, rmarkdown, covr, picante, rbenchmark, INLA, MCMCglmm, logistf, phylolm, ggplot2, ggridges, DHARMa, rr2

VignetteBuilder knitr

URL https://daijiang.github.io/phyr/,

https://github.com/daijiang/phyr/

BugReports https://github.com/daijiang/phyr/issues

Additional_repositories https://inla.r-inla-download.org/R/stable/

NeedsCompilation yes

Author Anthony Ives [aut], Russell Dinnage [aut] (<https://orcid.org/0000-0003-0846-2819>), Lucas A. Nell [aut] (<https://orcid.org/0000-0003-3209-0517>), Matthew Helmus [aut], Daijiang Li [aut, cre] (<https://orcid.org/0000-0002-0925-3421>)

Maintainer Daijiang Li <daijianglee@gmail.com>

Repository CRAN

Date/Publication 2020-12-18 07:30:05 UTC

Contents

`align_comm_V` *Create phylogenetic var-cov matrix based on phylogeny and community data*

Description

This function will remove species from community data that are not in the phylogeny. It will also remove tips from the phylogeny that are not in the community data. And then convert the phylogeny to a Var-cov matrix.

Usage

```
align_comm_V(comm, tree, prune.tree = FALSE, scale.vcv = TRUE)
```

Arguments

`comm`	A site by species data frame, with site names as row names.
`tree`	A phylogeny with "phylo" as class; or a phylogenetic var-covar matrix.
`prune.tree`	Whether to prune the tree first then use vcv.phylo function. Default is FALSE: use vcv.phylo first then subsetting the matrix.
`scale.vcv`	Whether to scale vcv to a correlation matrix.

Value

A list of the community data and the phylogenetic var-cov matrix.

`boot_ci` *Generic method to output bootstrap confidence intervals from an object.*

Description

Implemented only for `cor_phylo` objects thus far.

Usage

```
boot_ci(mod, ...)
```

Arguments

`mod`	A `cor_phylo` object.
`...`	Additional arguments.

Value

A list of confidence intervals.

`comm_a` *Example community data*

Description

A data frame with site names as row names, species names as column names, cells are the abundance of each species at each site.

Usage

`comm_a`

Format

A data frame with 15 sites and 15 species.

`comm_b` *Example community data*

Description

A data frame with site names as row names, species names as column names, cells are the abundance of each species at each site.

Usage

`comm_b`

Format

A data frame with 15 sites and 9 species.

`cor_phylo` *Correlations among multiple variates with phylogenetic signal*

Description

This function calculates Pearson correlation coefficients for multiple continuous variates that may have phylogenetic signal, allowing users to specify measurement error as the standard error of variate values at the tips of the phylogenetic tree. Phylogenetic signal for each variate is estimated from the data assuming that variate evolution is given by a Ornstein-Uhlenbeck process. Thus, the function allows the estimation of phylogenetic signal in multiple variates while incorporating correlations among variates. It is also possible to include independent variables (covariates) for each variate to remove possible confounding effects. `cor_phylo` returns the correlation matrix for variate values, estimates of phylogenetic signal for each variate, and regression coefficients for independent variables affecting each variate.

Usage

```
cor_phylo(variates, species, phy,
          covariates = NULL,
          meas_errors = NULL,
          data = sys.frame(sys.parent()),
          REML = TRUE,
          method = c("nelder-mead-r", "bobyqa",
              "subplex", "nelder-mead-nlopt", "sann"),
          no_corr = FALSE,
          constrain_d = FALSE,
          lower_d = 1e-7,
          rel_tol = 1e-6,
          max_iter = 1000,
          sann_options = NULL,
          verbose = FALSE,
          rcond_threshold = 1e-10,
          boot = 0,
          keep_boots = c("fail", "none", "all"))

## S3 method for class 'cor_phylo'
boot_ci(mod, refits = NULL, alpha = 0.05, ...)

## S3 method for class 'cor_phylo'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`variates`	A formula or a matrix specifying variates between which correlations are being calculated. The formula should be one-sided of the form `~ A + B + C` for variate vectors `A`, `B`, and `C` that are present in `data`. In the matrix case, the matrix must

	have `n` rows and `p` columns (for `p` variates); if the matrix columns aren't named, `cor_phylo` will name them `par_1 ... par_p`.
`species`	A one-sided formula implicating the variable inside `data` representing species, or a vector directly specifying the species. If a formula, it must be of the form `~ spp` for the `spp` object containing the species information inside `data`. If a vector, it must be the same length as that of the tip labels in `phy`, and it will be coerced to a character vector like `phy`'s tip labels.
`phy`	Either a phylogeny of class `phylo` or a prepared variance-covariance matrix. If it is a phylogeny, we will coerce tip labels to a character vector, and convert it to a variance-covariance matrix assuming brownian motion evolution. We will also standardize all var-cov matrices to have determinant of one.
`covariates`	A list specifying covariate(s) for each variate. The list can contain only two-sided formulas or matrices. Formulas should be of the typical form: `y ~ x1 + x2` or `y ~ x1 * x2`. If using a list of matrices, each item must be named (e.g., `list(y = matrix(...))` specifying variate `y`'s covariates). If the matrix columns aren't named, `cor_phylo` will name them `cov_1 ... cov_q`, where `q` is the total number of covariates for all variates. Having factor covariates is not supported. Defaults to `NULL`, which indicates no covariates.
`meas_errors`	A list or matrix containing standard errors for each variate. If a list, it must contain only two-sided formulas like those for `covariates` (except that you can't have multiple measurement errors for a single variate). You can additionally pass an `n`-row matrix with column names corresponding to the associated variate names. Defaults to `NULL`, which indicates no measurement errors.
`data`	An optional data frame, list, or environment that contains the variables in the model. By default, variables are taken from the environment from which `cor_phylo` was called.
`REML`	Whether REML (versus ML) should be used for model fitting. Defaults to `TRUE`.
`method`	Method of optimization using `nlopt` or `optim`. Options include `"nelder-mead-nlopt"`, `"bobyqa"`, `"subplex"`, `"nelder-mead-r"`, and `"sann"`. The first three are carried out by `nlopt`, and the latter two by `optim`. See `https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms/` for information on the `nlopt` algorithms. Defaults to `"nelder-mead-r"`.
`no_corr`	A single logical for whether to make all correlations zero. Running `cor_phylo` with `no_corr = TRUE` is useful for comparing it to the same model run with correlations != 0. Defaults to `FALSE`.
`constrain_d`	If `constrain_d` is `TRUE`, the estimates of `d` are constrained to be between zero and 1. This can make estimation more stable and can be tried if convergence is problematic. This does not necessarily lead to loss of generality of the results, because before using `cor_phylo`, branch lengths of `phy` can be transformed so that the "starter" tree has strong phylogenetic signal. Defaults to `FALSE`.
`lower_d`	Lower bound on the phylogenetic signal parameter. Defaults to `1e-7`.
`rel_tol`	A control parameter dictating the relative tolerance for convergence in the optimization. Defaults to `1e-6`.
`max_iter`	A control parameter dictating the maximum number of iterations in the optimization. Defaults to `1000`.

`sann_options`	A named list containing the control parameters for SANN minimization. This is only relevant if `method == "sann"`. This list can only contain the names `"maxit"`, `"temp"`, and/or `"tmax"`, which will control the maximum number of iterations, starting temperature, and number of function evaluations at each temperature, respectively. Defaults to `NULL`, which results in `maxit = 1000`, `temp = 1`, and `tmax = 1`. Note that these are different from the defaults for `optim`.
`verbose`	If `TRUE`, the model `logLik` and running estimates of the correlation coefficients and values of `d` are printed each iteration during optimization. Defaults to `FALSE`.
`rcond_threshold`	Threshold for the reciprocal condition number of two matrices inside the log likelihood function. Increasing this threshold makes the optimization process more strongly "bounce away" from badly conditioned matrices and can help with convergence and with estimates that are nonsensical. Defaults to `1e-10`.
`boot`	Number of parametric bootstrap replicates. Defaults to `0`.
`keep_boots`	Character specifying when to output data (indices, convergence codes, and simulated variate data) from bootstrap replicates. This is useful for troubleshooting when one or more bootstrap replicates fails to converge or outputs ridiculous results. Setting this to `"all"` keeps all `boot` parameter sets, `"fail"` keeps parameter sets from replicates that failed to converge, and `"none"` keeps no parameter sets. Defaults to `"fail"`.
`mod`	`cor_phylo` object that was run with the `boot` argument > 0.
`refits`	One or more `cp_refits` objects containing refits of `cor_phylo` bootstrap replicates. These are used when the original fit did not converge. Multiple `cp_refits` objects should be input as a list. For a given bootstrap replicate, the original fit's estimates will be used when the fit converged. If multiple `cp_refits` objects are input and more than one converged for a given replicate, the estimates from the first `cp_refits` object contain a converged fit for that replicate will be used. Defaults to `NULL`.
`alpha`	Alpha used for the confidence intervals. Defaults to `0.05`.
`...`	arguments passed to and from other methods.
`x`	an object of class `cor_phylo`.
`digits`	the number of digits to be printed.

Value

`cor_phylo` returns an object of class `cor_phylo`:

`call`	The matched call.
`corrs`	The `p x p` matrix of correlation coefficients.
`d`	Values of `d` from the OU process for each variate.
`B`	A matrix of regression-coefficient estimates, SE, Z-scores, and P-values, respectively. Rownames indicate which coefficient it refers to.
`B_cov`	Covariance matrix for regression coefficients.
`logLik`	The log likelihood for either the restricted likelihood (`REML = TRUE`) or the overall likelihood (`REML = FALSE`).

`AIC`	AIC for either the restricted likelihood (`REML = TRUE`) or the overall likelihood (`REML = FALSE`).
`BIC`	BIC for either the restricted likelihood (`REML = TRUE`) or the overall likelihood (`REML = FALSE`).
`niter`	Number of iterations the optimizer used.
`convcode`	Conversion code for the optimizer. This number is `0` on success and positive on failure. **1** iteration limit reached **2** generic failure code (nlopt optimizers only). **3** invalid arguments (nlopt optimizers only). **4** out of memory (nlopt optimizers only). **5** roundoff errors limited progress (nlopt optimizers only). **6** user-forced termination (nlopt optimizers only). **10** degeneracy of the Nelder-Mead simplex (`stats::optim` only). For more information on the nlopt return codes, see `https://nlopt.readthedocs.io/en/latest/NLopt_Reference/#return-values`.
`rcond_vals`	Reciprocal condition numbers for two matrices inside the log likelihood function. These are provided to potentially help guide the changing of the `rcond_threshold` parameter.
`bootstrap`	A list of bootstrap output, which is simply `list()` if `boot = 0`. If `boot > 0`, then the list contains fields for estimates of correlations (`corrs`), phylogenetic signals (`d`), coefficients (`B0`), and coefficient covariances (`B_cov`). It also contains the following information about the bootstrap replicates: a vector of indices relating each set of information to the bootstrapped estimates (`inds`), convergence codes (`convcodes`), and matrices of the bootstrapped parameters in the order they appear in the input argument (`mats`); these three fields will be empty if `keep_boots == "none"`. To view bootstrapped confidence intervals, use `boot_ci`.

`boot_ci` returns a list of confidence intervals with the following fields:

`corrs` Estimates of correlations. This is a matrix the values above the diagonal being the upper limits and values below being the lower limits.

`d` Phylogenetic signals.

`B0` Coefficient estimates.

`B_cov` Coefficient covariances.

Methods (by generic)

- `boot_ci`: returns bootstrapped confidence intervals from a `cor_phylo` object
- `print`: prints `cor_phylo` objects

Walkthrough

For the case of two variables, the function estimates parameters for the model of the form, for example,

$$X[1] = B[1,0] + B[1,1] * u[1,1] + \epsilon[1]$$
$$X[2] = B[2,0] + B[2,1] * u[2,1] + \epsilon[2]$$
$$\epsilon \; Gaussian(0, V)$$

where $B[1,0]$, $B[1,1]$, $B[2,0]$, and $B[2,1]$ are regression coefficients, and V is a variance-covariance matrix containing the correlation coefficient r, parameters of the OU process $d1$ and $d2$, and diagonal matrices $M1$ and $M2$ of measurement standard errors for $X[1]$ and $X[2]$. The matrix V is $2nx2n$, with nxn blocks given by

$$V[1,1] = C[1,1](d1) + M1$$
$$V[1,2] = C[1,2](d1, d2)$$
$$V[2,1] = C[2,1](d1, d2)$$
$$V[2,2] = C[2,2](d2) + M2$$

where $C[i,j](d1, d2)$ are derived from phy under the assumption of joint OU evolutionary processes for each variate (see Zheng et al. 2009). This formulation extends in the obvious way to more than two variates.

Author(s)

Anthony R. Ives, Lucas A. Nell

References

Zheng, L., A. R. Ives, T. Garland, B. R. Larget, Y. Yu, and K. F. Cao. 2009. New multivariate tests for phylogenetic signal and trait correlations applied to ecophysiological phenotypes of nine *Manglietia* species. *Functional Ecology* **23**:1059–1069.

Examples

```
#
# ## Simple example using data without correlations or phylogenetic
# ## signal. This illustrates the structure of the input data.
#
# set.seed(10)
# phy <- ape::rcoal(10, tip.label = 1:10)
# data_df <- data.frame(
#     species = phy$tip.label,
#     # variates:
#     par1 = rnorm(10),
#     par2 = rnorm(10),
#     par3 = rnorm(10),
```

```
#     # covariate for par2:
#     cov2 = rnorm(10, mean = 10, sd = 4),
#     # measurement error for par1 and par2, respectively:
#     se1 = 0.2,
#     se2 = 0.4
# )
# data_df$par2 <- data_df$par2 + 0.5 * data_df$cov2
#
#
# # cor_phylo(variates = ~ par1 + par2 + par3,
# #           covariates = list(par2 ~ cov2),
# #           meas_errors = list(par1 ~ se1, par2 ~ se2),
# #           species = ~ species,
# #           phy = phy,
# #           data = data_df)
#
# # If you've already created matrices/lists...
# X <- as.matrix(data_df[,c("par1", "par2", "par3")])
# U <- list(par2 = cbind(cov2 = data_df$cov2))
# M <- cbind(par1 = data_df$se1, par2 = data_df$se2)
#
# # ... you can also use those directly
# # (notice that I'm inputting an object for `species`
# # bc I ommitted `data`):
# # cor_phylo(variates = X, species = data_df$species,
# #           phy = phy, covariates = U,
# #           meas_errors = M)
#
#
#
#
# ## Simulation example for the correlation between two variables. The example
# ## compares the estimates of the correlation coefficients from cor_phylo when
# ## measurement error is incorporated into the analyses with three other cases:
# ## (i) when measurement error is excluded, (ii) when phylogenetic signal is
# ## ignored (assuming a "star" phylogeny), and (iii) neither measurement error
# ## nor phylogenetic signal are included.
#
# # In the simulations, variable 2 is associated with a single independent variable.
#
# library(ape)
#
# set.seed(1)
# # Set up parameter values for simulating data
# n <- 50
# phy <- rcoal(n, tip.label = 1:n)
# trt_names <- paste0("par", 1:2)
#
# R <- matrix(c(1, 0.7, 0.7, 1), nrow = 2, ncol = 2)
# d <- c(0.3, 0.95)
# B2 <- 1
#
# Se <- c(0.2, 1)
```

```
# M <- matrix(Se, nrow = n, ncol = 2, byrow = TRUE)
# colnames(M) <- trt_names
#
# # Set up needed matrices for the simulations
# p <- length(d)
#
# star <- stree(n)
# star$edge.length <- array(1, dim = c(n, 1))
# star$tip.label <- phy$tip.label
#
# Vphy <- vcv(phy)
# Vphy <- Vphy/max(Vphy)
# Vphy <- Vphy/exp(determinant(Vphy)$modulus[1]/n)
#
# tau <- matrix(1, nrow = n, ncol = 1) %*% diag(Vphy) - Vphy
# C <- matrix(0, nrow = p * n, ncol = p * n)
# for (i in 1:p) for (j in 1:p) {
#   Cd <- (d[i]^tau * (d[j]^t(tau)) * (1 - (d[i] * d[j])^Vphy))/(1 - d[i] * d[j])
#   C[(n * (i - 1) + 1):(i * n), (n * (j - 1) + 1):(j * n)] <- R[i, j] * Cd
# }
# MM <- matrix(M^2, ncol = 1)
# V <- C + diag(as.numeric(MM))
#
# # Perform a Cholesky decomposition of Vphy. This is used to generate phylogenetic
# # signal: a vector of independent normal random variables, when multiplied by the
# # transpose of the Cholesky deposition of Vphy will have covariance matrix
# # equal to Vphy.
# iD <- t(chol(V))
#
# # Perform Nrep simulations and collect the results
# Nrep <- 100
# cor.list <- matrix(0, nrow = Nrep, ncol = 1)
# cor.noM.list <- matrix(0, nrow = Nrep, ncol = 1)
# cor.noP.list <- matrix(0, nrow = Nrep, ncol = 1)
# cor.noMP.list <- matrix(0, nrow = Nrep, ncol = 1)
# d.list <- matrix(0, nrow = Nrep, ncol = 2)
# d.noM.list <- matrix(0, nrow = Nrep, ncol = 2)
# B.list <- matrix(0, nrow = Nrep, ncol = 3)
# B.noM.list <- matrix(0, nrow = Nrep, ncol = 3)
# B.noP.list <- matrix(0, nrow = Nrep, ncol = 3)
#
#
# set.seed(2)
# for (rep in 1:Nrep) {
#
#   XX <- iD %*% rnorm(2 * n)
#   X <- matrix(XX, n, p)
#   colnames(X) <- trt_names
#
#   U <- list(cbind(rnorm(n, mean = 2, sd = 10)))
#   names(U) <- trt_names[2]
#
#   X[,2] <- X[,2] + B2[1] * U[[1]][,1] - B2[1] * mean(U[[1]][,1])
```

```
#
#   # Call cor_phylo with (i) phylogeny and measurement error,
#   # (ii) just phylogeny,
#   # and (iii) just measurement error
#   z <- cor_phylo(variates = X,
#                  covariates = U,
#                  meas_errors = M,
#                  phy = phy,
#                  species = phy$tip.label)
#   z.noM <- cor_phylo(variates = X,
#                      covariates = U,
#                      phy = phy,
#                      species = phy$tip.label)
#   z.noP <- cor_phylo(variates = X,
#                      covariates = U,
#                      meas_errors = M,
#                      phy = star,
#                      species = phy$tip.label)
#
#   cor.list[rep] <- z$corrs[1, 2]
#   cor.noM.list[rep] <- z.noM$corrs[1, 2]
#   cor.noP.list[rep] <- z.noP$corrs[1, 2]
#   cor.noMP.list[rep] <- cor(cbind(
#     lm(X[,1] ~ 1)$residuals,
#     lm(X[,2] ~ U[[1]])$residuals))[1,2]
#
#   d.list[rep, ] <- z$d
#   d.noM.list[rep, ] <- z.noM$d
#
#   B.list[rep, ] <- z$B[,1]
#   B.noM.list[rep, ] <- z.noM$B[,1]
#   B.noP.list[rep, ] <- z.noP$B[,1]
# }
#
# correlation <- rbind(R[1, 2], mean(cor.list), mean(cor.noM.list),
#                      mean(cor.noP.list), mean(cor.noMP.list))
# rownames(correlation) <- c("True", "With M and Phy", "Without M",
#                            "Without Phy", "Without Phy or M")
#
# signal.d <- rbind(d, colMeans(d.list), colMeans(d.noM.list))
# rownames(signal.d) <- c("True", "With M and Phy", "Without M")
#
# est.B <- rbind(c(0, 0, B2), colMeans(B.list),
#                colMeans(B.noM.list[-39,]),  # 39th rep didn't converge
#                colMeans(B.noP.list))
# rownames(est.B) <- c("True", "With M and Phy", "Without M", "Without Phy")
# colnames(est.B) <- rownames(z$B)
#
# # Example simulation output:
#
# correlation
# #                       [,1]
# # True             0.7000000
```

```
# # With M and Phy   0.6943712
# # Without M        0.2974162
# # Without Phy      0.3715406
# # Without Phy or M 0.3291473
#
# signal.d
# #                    [,1]      [,2]
# # True           0.3000000 0.9500000
# # With M and Phy 0.3025853 0.9422067
# # Without M      0.2304527 0.4180208
#
# est.B
# #                      par1_0    par2_0 par2_cov_1
# # True            0.000000000 0.0000000  1.0000000
# # With M and Phy -0.008838245 0.1093819  0.9995058
# # Without M      -0.008240453 0.1142330  0.9995625
# # Without Phy     0.002933341 0.1096578  1.0028474
```

`envi` *Example environmental data*

Description

A data frame of site environmental variables.

Usage

```
envi
```

Format

A data frame with 15 sites and 4 variables: sand proportion, canopy shade proportion, precipitation, and minimum temperature.

`family.communityPGLMM` *Family Objects for communityPGLMM objects*

Description

Family Objects for communityPGLMM objects

Usage

```
## S3 method for class 'communityPGLMM'
family(object, ...)
```

Arguments

`object`	the function `family` accesses the `family` objects which are stored within objects created by modelling functions (e.g., `glm`).
`...`	further arguments passed to methods.

`fitted.communityPGLMM` *Fitted values for communityPGLMM*

Description

Fitted values for communityPGLMM

Usage

```
## S3 method for class 'communityPGLMM'
fitted(object, ...)
```

Arguments

`object`	A fitted model with class communityPGLMM.
`...`	Additional arguments, ignored for method compatibility.

Value

Fitted values. For binomial and poisson PGLMMs, this is equal to mu.

`fixef` *Extract fixed-effects estimates*

Description

Extract the fixed-effects estimates

Usage

```
## S3 method for class 'communityPGLMM'
fixef(object, ...)
```

Arguments

`object`	A fitted model with class communityPGLMM.
`...`	Ignored.

Details

Extract the estimates of the fixed-effects parameters from a fitted model. For bayesian models, the p-values are simply to indicate whether the credible intervals include 0 (p = 0.04) or not (p = 0.6).

Value

A dataframe of fixed-effects estimates.

`get_design_matrix` `get_design_matrix` *gets design matrix for gaussian, binomial, and poisson models*

Description

`get_design_matrix` gets design matrix for gaussian, binomial, and poisson models

Usage

```
get_design_matrix(formula, data, random.effects, na.action = NULL)
```

Arguments

`formula`	A two-sided linear formula object describing the mixed effects of the model. To specify that a random term should have phylogenetic covariance matrix along with non-phylogenetic one, add `__` (two underscores) at the end of the group variable; e.g., `+ (1 | sp__)` will construct two random terms, one with phylogenetic covariance matrix and another with non-phylogenetic (identity) matrix. In contrast, `__` in the nested terms (below) will only create a phylogenetic covariance matrix. Nested random terms have the general form `(1|sp__@site__)` which represents phylogenetically related species nested within correlated sites. This form can be used for bipartite questions. For example, species could be phylogenetically related pollinators and sites could be phylogenetically related plants, leading to the random effect `(1|insects__@plants__)`. If more than one phylogeny is used, remember to add all to the argument `cov_ranef = list(insects = insect_phylo, plants = plant_phylo)`. Phylogenetic correlations can be dropped by removing the `__` underscores. Thus, the form `(1|sp@site__)` excludes the phylogenetic correlations among species, while the form `(1|sp__@site)` excludes the correlations among sites. Note that correlated random terms are not allowed. For example, `(x|g)` will be the same as `(0 + x|g)` in the `lme4::lmer` syntax. However, `(x1 + x2|g)` won't work, so instead use `(x1|g) + (x2|g)`.

`data`	A `data.frame` containing the variables named in formula.
`random.effects`	Optional pre-build list of random effects. If `NULL` (the default), the function `prep_dat_pglmm` will prepare the random effects for you from the information in `formula`, `data`, and `cov_ranef`. `random.effect` allows a list of pre-generated random effects terms to increase flexibility; for example, this makes it possible to construct models with both phylogenetic correlation and spatio-temporal autocorrelation. In preparing `random.effect`, make sure that the orders of rows and columns of covariance matrices in the list are the same as their corresponding group variables in the data. Also, this should be *a list of lists*, e.g. `random.effects = list(re1 = list(matrix_a), re2 = list(1, sp = sp, covar = Vsp))`.
`na.action`	What to do with NAs?

Value

A list of design matrices.

`match_comm_tree` *Match phylogeny with community data*

Description

This function will remove species from community data that are not in the phylogeny. It will also remove tips from the phylogeny that are not in the community data.

Usage

```
match_comm_tree(comm, tree, comm_2 = NULL)
```

Arguments

`comm`	A site by species data frame, with site names as row names.
`tree`	A phylogeny with "phylo" as class.
`comm_2`	Another optional site by species data frame, if presented, both community data and the phylogeny will have the same set of species. This can be useful for PCD with custom species pool.

Value

A list of the community data and the phylogeny.

`model.frame.communityPGLMM` *Extracting the Model Frame from a communityPGLMM Model object*

Description

Extracting the Model Frame from a communityPGLMM Model object

Usage

```
## S3 method for class 'communityPGLMM'
model.frame(formula, ...)
```

Arguments

`formula`	a model `formula` or `terms` object or an R object.
`...`	for `model.frame` methods, a mix of further arguments such as `data`, `na.action`, `subset` to pass to the default method. Any additional arguments (such as `offset` and `weights` or other named arguments) which reach the default method are used to create further columns in the model frame, with parenthesised names such as `"(offset)"`. For `get_all_vars`, further named columns to include in the model frame.

`nobs.communityPGLMM` *Number of Observation in a communityPGLMM Model*

Description

Number of Observation in a communityPGLMM Model

Usage

```
## S3 method for class 'communityPGLMM'
nobs(object, use.fallback = FALSE, ...)
```

Arguments

`object`	A fitted model object.
`use.fallback`	logical: should fallback methods be used to try to guess the value?
`...`	Further arguments to be passed to methods.

`oldfield` *Phylogeny and community data from an Oldfield ecosystem in Southern Ontario, Canada*

Description

A list containing a phylogeny for XX species of Oldfield forbs, as well as a presence / absence dataset for their occurrence across several locations in Southern Ontario see Dinnage (2009) for details. Sites each had two plots which experienced a different treatment each; either they has been disturbed (ploughed 1 or 2 years previously), or they were a control plot (undisturbed in recent records).

Usage

```
oldfield
```

Format

A list with two elements:

- `phy` A phylogeny in `ape`'s `phy` format
- `data` A data.frame containing data on the occurrence of the species in `phy`

oldfield$data is a data.frame with 1786 rows, and the following 7 columns:

- `site_orig` integer. Site ID number.
- `habitat_type` character. Plot treatment: disturbed or undisturbed.
- `sp` character. Species name using underscore to separate binomial names (to match phylogeny).
- `abundance` integer. Recorded abundance of species in plot.
- `disturbance` integer. Whether the plot was disturbed or not. 0 or 1. 0 for undisturbed, 1 for disturbed
- `site_orig` character. A unique site descriptor concatenating the site number with the disturbance treatment.
- `pres` integer. Species presence or absence in plot. 0 or 1. 0 for absent, 1 for present

---

`pcd` *pairwise site phylogenetic community dissimilarity (PCD) within a community*

Description

Calculate pairwise site PCD, users can specify expected values from `pcd_pred()`.

Usage

```
pcd(comm, tree, expectation = NULL, cpp = TRUE, verbose = TRUE, ...)
```

Arguments

`comm`	A site by species data frame or matrix, sites as rows.
`tree`	A phylogeny for species.
`expectation`	nsp_pool, psv_bar, psv_pool, and nsr calculated from `pcd_pred()`.
`cpp`	Whether to use loops written with c++, default is TRUE.
`verbose`	Do you want to see the progress?
`...`	Other arguments.

Value

A list of a variety of pairwise dissimilarities.

References

Ives, A. R., & Helmus, M. R. 2010. Phylogenetic metrics of community similarity. The American Naturalist, 176(5), E128-E142.

Examples

```
x1 = pcd_pred(comm_1 = comm_a, comm_2 = comm_b, tree = phylotree, reps = 100)
pcd(comm = comm_a, tree = phylotree, expectation = x1)
```

`pcd_pred` *Predicted PCD with species pool*

Description

This function will calculate expected PCD from one or two sets of communities (depends on the species pool)

Usage

```
pcd_pred(comm_1, comm_2 = NULL, tree, reps = 10^3, cpp = TRUE)
```

Arguments

`comm_1`	A site by species dataframe or matrix, with sites as rows and species as columns.
`comm_2`	An optional second site by species data frame. It should have the same number of rows as comm_1. This can be useful if we want to calculate temporal beta diversity, i.e. changes of the same site over time. Because data of the same site are not independent, setting comm_2 will use both communities as species pool to calculate expected PCD.
`tree`	The phylogeny for all species, with "phylo" as class; or a var-cov matrix.
`reps`	Number of random draws, default is 1000 times.
`cpp`	Whether to use loops written with c++, default is TRUE. If you came across with errors, try to set cpp = FALSE. This normally will run without errors, but slower.

Value

A list with species richness of the pool, expected PSV, PSV of the pool, and unique number of species richness across sites.

`pglmm` *Phylogenetic Generalized Linear Mixed Model for Community Data*

Description

This function performs Generalized Linear Mixed Models for binary, count, and continuous data, estimating regression coefficients with approximate standard errors. It is specifically designed for community data in which species occur within multiple sites (locations). A Bayesian version of PGLMM uses the package `INLA`, which is not available on CRAN yet. If you wish to use this option, you must first install `INLA` from https://www.r-inla.org/ by running `install.packages('INLA', repos='https://www.math.ntnu.no/inla/R/stable')` in R.

Usage

```
pglmm(
  formula,
  data = NULL,
  family = "gaussian",
  cov_ranef = NULL,
  random.effects = NULL,
  REML = TRUE,
  optimizer = c("nelder-mead-nlopt", "bobyqa", "Nelder-Mead", "subplex"),
  repulsion = FALSE,
  add.obs.re = TRUE,
  verbose = FALSE,
  cpp = TRUE,
  bayes = FALSE,
  s2.init = NULL,
  B.init = NULL,
  reltol = 10^-6,
  maxit = 500,
  tol.pql = 10^-6,
  maxit.pql = 200,
  marginal.summ = "mean",
  calc.DIC = TRUE,
  calc.WAIC = TRUE,
  prior = "inla.default",
  prior_alpha = 0.1,
  prior_mu = 1,
  ML.init = FALSE,
  tree = NULL,
  tree_site = NULL,
  sp = NULL,
  site = NULL,
  bayes_options = NULL,
  bayes_nested_matrix_as_list = FALSE
)

communityPGLMM(
  formula,
  data = NULL,
  family = "gaussian",
  cov_ranef = NULL,
  random.effects = NULL,
  REML = TRUE,
  optimizer = c("nelder-mead-nlopt", "bobyqa", "Nelder-Mead", "subplex"),
  repulsion = FALSE,
  add.obs.re = TRUE,
  verbose = FALSE,
  cpp = TRUE,
  bayes = FALSE,
```

```
  s2.init = NULL,
  B.init = NULL,
  reltol = 10^-6,
  maxit = 500,
  tol.pql = 10^-6,
  maxit.pql = 200,
  marginal.summ = "mean",
  calc.DIC = TRUE,
  calc.WAIC = TRUE,
  prior = "inla.default",
  prior_alpha = 0.1,
  prior_mu = 1,
  ML.init = FALSE,
  tree = NULL,
  tree_site = NULL,
  sp = NULL,
  site = NULL,
  bayes_options = NULL,
  bayes_nested_matrix_as_list = FALSE
)
```

Arguments

`formula`	A two-sided linear formula object describing the mixed effects of the model. To specify that a random term should have phylogenetic covariance matrix along with non-phylogenetic one, add `__` (two underscores) at the end of the group variable; e.g., `+ (1 | sp__)` will construct two random terms, one with phylogenetic covariance matrix and another with non-phylogenetic (identity) matrix. In contrast, `__` in the nested terms (below) will only create a phylogenetic covariance matrix. Nested random terms have the general form `(1|sp__@site__)` which represents phylogenetically related species nested within correlated sites. This form can be used for bipartite questions. For example, species could be phylogenetically related pollinators and sites could be phylogenetically related plants, leading to the random effect `(1|insects__@plants__)`. If more than one phylogeny is used, remember to add all to the argument `cov_ranef = list(insects = insect_phylo, plants = plant_phylo)`. Phylogenetic correlations can be dropped by removing the `__` underscores. Thus, the form `(1|sp@site__)` excludes the phylogenetic correlations among species, while the form `(1|sp__@site)` excludes the correlations among sites. Note that correlated random terms are not allowed. For example, `(x|g)` will be the same as `(0 + x|g)` in the `lme4::lmer` syntax. However, `(x1 + x2|g)` won't work, so instead use `(x1|g) + (x2|g)`.
`data`	A `data.frame` containing the variables named in formula.
`family`	Either "gaussian" for a Linear Mixed Model, or "binomial" or "poisson" for Generalized Linear Mixed Models. "family" should be specified as a character string (i.e., quoted). For binomial and Poisson data, we use the canonical logit and log link functions, respectively. Binomial data can be either presence/absence, or

	a two-column array of 'successes' and 'failures'. For both binomial and Poisson data, we add an observation-level random term by default via `add.obs.re = TRUE`. If `bayes = TRUE` there are two additional families available: "zeroinflated.binomial", and "zeroinflated.poisson", which add a zero inflation parameter; this parameter gives the probability that the response is a zero. The rest of the parameters of the model then reflect the "non-zero" part part of the model. Note that "zeroinflated.binomial" only makes sense for success/failure response data.
`cov_ranef`	A named list of covariance matrices of random terms. The names should be the group variables that are used as random terms with specified covariance matrices (without the two underscores, e.g. `list(sp = tree1, site = tree2)`). The actual object can be either a phylogeny with class "phylo" or a prepared covariance matrix. If it is a phylogeny, `pglmm` will prune it and then convert it to a covariance matrix assuming Brownian motion evolution. `pglmm` will also standardize all covariance matrices to have determinant of one. Group variables will be converted to factors and all covariance matrices will be rearranged so that rows and columns are in the same order as the levels of their corresponding group variables.
`random.effects`	Optional pre-build list of random effects. If `NULL` (the default), the function `prep_dat_pglmm` will prepare the random effects for you from the information in `formula`, `data`, and `cov_ranef`. `random.effect` allows a list of pre-generated random effects terms to increase flexibility; for example, this makes it possible to construct models with both phylogenetic correlation and spatio-temporal autocorrelation. In preparing `random.effect`, make sure that the orders of rows and columns of covariance matrices in the list are the same as their corresponding group variables in the data. Also, this should be *a list of lists*, e.g. `random.effects = list(re1 = list(matrix_a), re2 = list(1, sp = sp, covar = Vsp))`.
`REML`	Whether REML or ML is used for model fitting the random effects. Ignored if `bayes = TRUE`.
`optimizer`	nelder-mead-nlopt (default), bobyqa, Nelder-Mead, or subplex. Nelder-Mead is from the stats package and the other optimizers are from the nloptr package. Ignored if `bayes = TRUE`.
`repulsion`	When there are nested random terms specified, `repulsion = FALSE` tests for phylogenetic underdispersion while `repulsion = FALSE` tests for overdispersion. This argument is a logical vector of length either 1 or >1. If its length is 1, then all covariance matrices in nested terms will be either inverted (overdispersion) or not. If its length is >1, then you can select which covariance matrix in the nested terms to be inverted. Make sure to get the length right: for all the terms with `@`, count the number of "__" to determine the length of repulsion. For example, `sp__@site` and `sp@site__` will each require one element of `repulsion`, while `sp__@site__` will take two elements (repulsion for sp and repulsion for site). Therefore, if your nested terms are `(1|sp__@site) + (1|sp@site__) + (1|sp__@site__)`, then you should set the repulsion to be something like `c(TRUE, FALSE, TRUE, TRUE)` (length of 4).
`add.obs.re`	Whether to add an observation-level random term for binomial or Poisson distributions. Normally it would be a good idea to add this to account for overdis-

	persion, so `add.obs.re = TRUE` by default.
`verbose`	If `TRUE`, the model deviance and running estimates of `s2` and `B` are plotted each iteration during optimization.
`cpp`	Whether to use C++ function for optim. Default is TRUE. Ignored if `bayes = TRUE`.
`bayes`	Whether to fit a Bayesian version of the PGLMM using `r-inla`.
`s2.init`	An array of initial estimates of s2 for each random effect that scales the variance. If s2.init is not provided for `family="gaussian"`, these are estimated using `lm` assuming no phylogenetic signal. A better approach might be to run `link[lme4:lmer]{lmer}` and use the output random effects for `s2.init`. If `s2.init` is not provided for `family = "binomial"`, these are set to 0.25.
`B.init`	Initial estimates of B, a matrix containing regression coefficients in the model for the fixed effects. This matrix must have `dim(B.init) = c(p + 1, 1)`, where `p` is the number of predictor (independent) variables; the first element of `B` corresponds to the intercept, and the remaining elements correspond in order to the predictor (independent) variables in the formula. If `B.init` is not provided, these are estimated using `lm` or `glm` assuming no phylogenetic signal. A better approach might be to run `lmer` and use the output fixed effects for `B.init`. When `bayes = TRUE`, initial values are estimated using the maximum likelihood fit unless `ML.init = FALSE`, in which case the default `INLA` initial values will be used.
`reltol`	A control parameter dictating the relative tolerance for convergence in the optimization; see `optim`.
`maxit`	A control parameter dictating the maximum number of iterations in the optimization; see `optim`.
`tol.pql`	A control parameter dictating the tolerance for convergence in the PQL estimates of the mean components of the GLMM. Ignored if `family = "gaussian"` or `bayes = TRUE`.
`maxit.pql`	A control parameter dictating the maximum number of iterations in the PQL estimates of the mean components of the GLMM. Ignored if `family = "gaussian"` or `bayes = TRUE`.
`marginal.summ`	Summary statistic to use for the estimate of coefficients when doing a Bayesian PGLMM (when `bayes = TRUE`). Options are: "mean", "median", or "mode", referring to different characterizations of the central tendency of the Bayesian posterior marginal distributions. Ignored if `bayes = FALSE`.
`calc.DIC`	Should the Deviance Information Criterion be calculated and returned when doing a Bayesian PGLMM? Ignored if `bayes = FALSE`.
`calc.WAIC`	Should the WAIC be calculated and returned when doing a Bayesian PGLMM? Ignored if `bayes = FALSE`.
`prior`	Which type of default prior should be used by `pglmm`? Only used if `bayes = TRUE`. There are currently four options: "inla.default", which uses the default `INLA` priors; "pc.prior.auto", which uses a complexity penalizing prior (as described in Simpson et al. (2017)) designed to automatically choose good parameters (only available for gaussian and binomial responses); "pc.prior", which allows the user to set custom parameters on the "pc.prior" prior, using the

	`prior_alpha` and `prior_mu` parameters (Run `INLA::inla.doc("pc.prec")` for details on these parameters); and "uninformative", which sets a very uninformative prior (nearly uniform) by using a very flat exponential distribution. The last option is generally not recommended but may in some cases give estimates closer to the maximum likelihood estimates. "pc.prior.auto" is only implemented for `family = "gaussian"` and `family = "binomial"` currently.
`prior_alpha`	Only used if `bayes = TRUE` and `prior = "pc.prior"`, in which case it sets the alpha parameter of `INLA`'s complexity penalizing prior for the random effects. The prior is an exponential distribution where prob(sd > mu) = alpha, where sd is the standard deviation of the random effect.
`prior_mu`	Only used if `bayes = TRUE` and `prior = "pc.prior"`, in which case it sets the mu parameter of `INLA`'s complexity penalizing prior for the random effects. The prior is an exponential distribution where prob(sd > mu) = alpha, where sd is the standard deviation of the random effect.
`ML.init`	Only relevant if `bayes = TRUE`. Should maximum likelihood estimates be calculated and used as initial values for the Bayesian model fit? Sometimes this can be helpful, but it may not help; thus, we set the default to `FALSE`. Also, it does not work with the zero-inflated families.
`tree`	A phylogeny for column sp, with "phylo" class, or a covariance matrix for sp. Make sure to have all species in the matrix; if the matrix is not standardized, (i.e., det(tree) != 1), `pglmm` will try to standardize it for you. No longer used: keep here for compatibility.
`tree_site`	A second phylogeny for "site". This is required only if the site column contains species instead of sites. This can be used for bipartitie questions; tree_site can also be a covariance matrix. Make sure to have all sites in the matrix; if the matrix is not standardized (i.e., det(tree_site) != 1), pglmm‘ will try to standardize it for you. No longer used: keep here for compatibility.
`sp`	No longer used: keep here for compatibility.
`site`	No longer used: keep here for compatibility.
`bayes_options`	Additional options to pass to INLA for if `bayes = TRUE`. A named list where the names correspond to parameters in the `inla` function. One special option is `diagonal`: if an element in the options list is names `diagonal` this tells `INLA` to add its value to the diagonal of the random effects precision matrices. This can help with numerical stability if the model is ill-conditioned (if you get a lot of warnings, try setting this to `list(diagonal = 1e-4)`).
`bayes_nested_matrix_as_list`	For `bayes = TRUE`, prepare the nested terms as a list of length of 4 as the old way?

Details

For Gaussian data, `pglmm` analyzes the phylogenetic linear mixed model

$$Y = \beta_0 + \beta_1 x + b_0 + b_1 x$$

$$b_0 \; Gaussian(0, \sigma_0^2 I_{sp})$$

$$b_1 \; Gaussian(0, \sigma_0^2 V_{sp})$$
$$\eta \; Gaussian(0, \sigma^2)$$

where β_0 and β_1 are fixed effects, and V_{sp} is a variance-covariance matrix derived from a phylogeny (typically under the assumption of Brownian motion evolution). Here, the variation in the mean (intercept) for each species is given by the random effect b_0 that is assumed to be independent among species. Variation in species' responses to predictor variable x is given by a random effect b_0 that is assumed to depend on the phylogenetic relatedness among species given by V_{sp}; if species are closely related, their specific responses to x will be similar. This particular model would be specified as

`z <- pglmm(Y ~ X + (1|sp__), data = data, family = "gaussian", cov_ranef = list(sp = phy))`

Or you can prepare the random terms manually (not recommended for simple models but may be necessary for complex models):

`re.1 <- list(1, sp = dat$sp, covar = diag(nspp))`

`re.2 <- list(dat$X, sp = dat$sp, covar = Vsp)`

`z <- pglmm(Y ~ X, data = data, family = "gaussian", random.effects = list(re.1, re.2))`

The covariance matrix covar is standardized to have its determinant equal to 1. This in effect standardizes the interpretation of the scalar σ^2. Although mathematically this is not required, it is a very good idea to standardize the predictor (independent) variables to have mean 0 and variance 1. This will make the function more robust and improve the interpretation of the regression coefficients. For categorical (factor) predictor variables, you will need to construct 0-1 dummy variables, and these should not be standardized (for obvious reasons).

For binary generalized linear mixed models (`family = 'binomial'`), the function estimates parameters for the model of the form, for example,

$$y = \beta_0 + \beta_1 x + b_0 + b_1 x$$
$$Y = logit^{-1}(y)$$
$$b_0 \; Gaussian(0, \sigma_0^2 I_{sp})$$
$$b_1 \; Gaussian(0, \sigma_0^2 V_{sp})$$

where β_0 and β_1 are fixed effects, and V_{sp} is a variance-covariance matrix derived from a phylogeny (typically under the assumption of Brownian motion evolution).

`z <- pglmm(Y ~ X + (1|sp__), data = data, family = "binomial", cov_ranef = list(sp = phy))`

As with the linear mixed model, it is a very good idea to standardize the predictor (independent) variables to have mean 0 and variance 1. This will make the function more robust and improve the interpretation of the regression coefficients.

Value

An object (list) of class `communityPGLMM` with the following elements:

`formula`	the formula for fixed effects
`formula_original`	the formula for both fixed effects and random effects

`data`	the dataset
`family`	`gaussian`, `binomial`, or `poisson` depending on the model fit
`random.effects`	the list of random effects
`B`	estimates of the regression coefficients
`B.se`	approximate standard errors of the fixed effects regression coefficients. This is set to NULL if `bayes = TRUE`.
`B.ci`	approximate Bayesian credible interval of the fixed effects regression coefficients. This is set to NULL if `bayes = FALSE`
`B.cov`	approximate covariance matrix for the fixed effects regression coefficients
`B.zscore`	approximate Z scores for the fixed effects regression coefficients. This is set to NULL if `bayes = TRUE`
`B.pvalue`	approximate tests for the fixed effects regression coefficients being different from zero. This is set to NULL if `bayes = TRUE`
`ss`	standard deviations of the random effects for the covariance matrix $\sigma^2 V$ for each random effect in order. For the linear mixed model, the residual variance is listed last.
`s2r`	random effects variances for non-nested random effects
`s2n`	random effects variances for nested random effects
`s2resid`	for linear mixed models, the residual variance
`s2r.ci`	Bayesian credible interval for random effects variances for non-nested random effects. This is set to NULL if `bayes = FALSE`
`s2n.ci`	Bayesian credible interval for random effects variances for nested random effects. This is set to NULL if `bayes = FALSE`
`s2resid.ci`	Bayesian credible interval for linear mixed models, the residual variance. This is set to NULL if `bayes = FALSE`
`logLik`	for linear mixed models, the log-likelihood for either the restricted likelihood (`REML=TRUE`) or the overall likelihood (`REML=FALSE`). This is set to NULL for generalized linear mixed models. If `bayes = TRUE`, this is the marginal log-likelihood
`AIC`	for linear mixed models, the AIC for either the restricted likelihood (`REML = TRUE`) or the overall likelihood (`REML = FALSE`). This is set to NULL for generalised linear mixed models
`BIC`	for linear mixed models, the BIC for either the restricted likelihood (`REML = TRUE`) or the overall likelihood (`REML = FALSE`). This is set to NULL for generalised linear mixed models
`DIC`	for Bayesian PGLMM, this is the Deviance Information Criterion metric of model fit. This is set to NULL if `bayes = FALSE`.
`REML`	whether or not REML is used (`TRUE` or `FALSE`).
`bayes`	whether or not a Bayesian model was fit.
`marginal.summ`	The specified summary statistic used to summarize the Bayesian marginal distributions. Only present if `bayes = TRUE`

`s2.init`	the user-provided initial estimates of `s2`
`B.init`	the user-provided initial estimates of `B`
`Y`	the response (dependent) variable returned in matrix form
`X`	the predictor (independent) variables returned in matrix form (including 1s in the first column)
`H`	the residuals. For linear mixed models, this does not account for random terms, To get residuals after accounting for both fixed and random terms, use `residuals()`. For the generalized linear mixed model, these are the predicted residuals in the logit -1 space.
`iV`	the inverse of the covariance matrix for the entire system (of dimension (`nsp` * `nsite`) by (`nsp` * `nsite`)). This is NULL if `bayes = TRUE`.
`mu`	predicted mean values for the generalized linear mixed model (i.e., similar to `fitted(merMod)`). Set to NULL for linear mixed models, for which we can use `fitted()`.
`nested`	matrices used to construct the nested design matrix. This is set to NULL if `bayes = TRUE`
`Zt`	the design matrix for random effects. This is set to NULL if `bayes = TRUE`
`St`	diagonal matrix that maps the random effects variances onto the design matrix
`convcode`	the convergence code provided by `optim`. This is set to NULL if `bayes = TRUE`
`niter`	number of iterations performed by `optim`. This is set to NULL if `bayes = TRUE`
`inla.model`	Model object fit by underlying `inla` function. Only returned if `bayes = TRUE`

Author(s)

Anthony R. Ives, Daijiang Li, Russell Dinnage

References

Ives, A. R. and M. R. Helmus. 2011. Generalized linear mixed models for phylogenetic analyses of community structure. Ecological Monographs 81:511-525.

Ives A. R. 2018. Mixed and phylogenetic models: a conceptual introduction to correlated data. https://leanpub.com/correlateddata.

Rafferty, N. E., and A. R. Ives. 2013. Phylogenetic trait-based analyses of ecological networks. Ecology 94:2321-2333.

Simpson, Daniel, et al. 2017. Penalising model component complexity: A principled, practical approach to constructing priors. Statistical science 32(1): 1-28.

Li, D., Ives, A. R., & Waller, D. M. 2017. Can functional traits account for phylogenetic signal in community composition? New Phytologist, 214(2), 607-618.

Examples

```
## Structure of examples:
# First, a (brief) description of model types, and how they are specified
# - these are *not* to be run 'as-is'; they show how models should be organised
```

```
# Second, a run-through of how to simulate, and then analyse, data
# - these *are* to be run 'as-is'; they show how to format and work with data


#############################################
### Brief summary of models and their use ###
#############################################
## Model structures from Ives & Helmus (2011)
if(FALSE){
  # dat = data set for regression (note: must have a column "sp" and a column "site")
  # phy = phylogeny of class "phylo"
  # repulsion = to test phylogenetic repulsion or not

  # Model 1 (Eq. 1)
  z <- pglmm(freq ~ sp + (1|site) + (1|sp__@site), data = dat, family = "binomial",
             cov_ranef = list(sp = phy), REML = TRUE, verbose = TRUE, s2.init = .1)

  # Model 2 (Eq. 2)
  z <- pglmm(freq ~ sp + X + (1|site) + (X|sp__), data = dat, family = "binomial",
             cov_ranef = list(sp = phy), REML = TRUE, verbose = TRUE, s2.init = .1)

  # Model 3 (Eq. 3)
  z <- pglmm(freq ~ sp*X + (1|site) + (1|sp__@site), data = dat, family = "binomial",
             cov_ranef = list(sp = phy), REML = TRUE, verbose = TRUE, s2.init = .1)

  ## Model structure from Rafferty & Ives (2013) (Eq. 3)
  # dat = data set
  # phyPol = phylogeny for pollinators (pol)
  # phyPlt = phylogeny for plants (plt)

  z <- pglmm(freq ~ pol * X + (1|pol__) + (1|plt__) + (1|pol__@plt) +
               (1|pol@plt__) + (1|pol__@plt__),
             data = dat, family = "binomial",
             cov_ranef = list(pol = phyPol, plt = phyPlt),
             REML = TRUE, verbose = TRUE, s2.init = .1)
}

#####################################################
### Detailed analysis showing covariance matrices ###
#####################################################

# This is the example from section 4.3 in Ives, A. R. (2018) Mixed
# and phylogenetic models: a conceptual introduction to correlated data.

library(ape)
library(mvtnorm)

# Investigating covariance matrices for different types of model structure
nspp <- 6
nsite <- 4

# Simulate a phylogeny that has a lot of phylogenetic signal (power = 1.3)
phy <- compute.brlen(rtree(n = nspp), method = "Grafen", power = 1.3)
```

```
# Simulate species means
sd.sp <- 1
mean.sp <- rTraitCont(phy, model = "BM", sigma=sd.sp^2)

# Replicate values of mean.sp over sites
Y.sp <- rep(mean.sp, times=nsite)

# Simulate site means
sd.site <- 1
mean.site <- rnorm(nsite, sd=sd.site)

# Replicate values of mean.site over sp
Y.site <- rep(mean.site, each=nspp)

# Compute a covariance matrix for phylogenetic attraction
sd.attract <- 1
Vphy <- vcv(phy)

# Standardize the phylogenetic covariance matrix to have determinant = 1.
# (For an explanation of this standardization, see subsection 4.3.1 in Ives (2018))
Vphy <- Vphy/(det(Vphy)^(1/nspp))

# Construct the overall covariance matrix for phylogenetic attraction.
# (For an explanation of Kronecker products, see subsection 4.3.1 in the book)
V <- kronecker(diag(nrow = nsite, ncol = nsite), Vphy)
Y.attract <- array(t(rmvnorm(n = 1, sigma = sd.attract^2*V)))

# Simulate residual errors
sd.e <- 1
Y.e <- rnorm(nspp*nsite, sd = sd.e)

# Construct the dataset
d <- data.frame(sp = rep(phy$tip.label, times = nsite),
                site = rep(1:nsite, each = nspp))

# Simulate abundance data
d$Y <- Y.sp + Y.site + Y.attract + Y.e

# Analyze the model
pglmm(Y ~ 1 + (1|sp__) + (1|site) + (1|sp__@site), data = d, cov_ranef = list(sp = phy))

# Display random effects: the function `pglmm_plot_ranef()` does what
# the name implies. You can set `show.image = TRUE` and `show.sim.image = TRUE`
# to see the matrices and simulations.
re <- pglmm_plot_ranef(Y ~ 1 + (1|sp__) + (1|site) + (1|sp__@site), data = d,
                    cov_ranef = list(sp = phy), show.image = FALSE,
                    show.sim.image = FALSE)

#################################################
### Example of a bipartite phylogenetic model ###
#################################################
```

```
# Investigating covariance matrices for different types of model structure
nspp <- 20
nsite <- 15

# Simulate a phylogeny that has a lot of phylogenetic signal (power = 1.3)
phy.sp <- compute.brlen(rtree(n = nspp), method = "Grafen", power = 1.3)
phy.site <- compute.brlen(rtree(n = nsite), method = "Grafen", power = 1.3)

# Simulate species means
mean.sp <- rTraitCont(phy.sp, model = "BM", sigma = 1)

# Replicate values of mean.sp over sites
Y.sp <- rep(mean.sp, times = nsite)

# Simulate site means
mean.site <- rTraitCont(phy.site, model = "BM", sigma = 1)

# Replicate values of mean.site over sp
Y.site <- rep(mean.site, each = nspp)

# Generate covariance matrix for phylogenetic attraction among species
sd.sp.attract <- 1
Vphy.sp <- vcv(phy.sp)
Vphy.sp <- Vphy.sp/(det(Vphy.sp)^(1/nspp))
V.sp <- kronecker(diag(nrow = nsite, ncol = nsite), Vphy.sp)
Y.sp.attract <- array(t(rmvnorm(n = 1, sigma = sd.sp.attract^2*V.sp)))
# Generate covariance matrix for phylogenetic attraction among sites
sd.site.attract <- 1
Vphy.site <- vcv(phy.site)
Vphy.site <- Vphy.site/(det(Vphy.site)^(1/nsite))
V.site <- kronecker(Vphy.site, diag(nrow = nspp, ncol = nspp))
Y.site.attract <- array(t(rmvnorm(n = 1, sigma = sd.site.attract^2*V.site)))

# Generate covariance matrix for phylogenetic attraction of species:site interaction
sd.sp.site.attract <- 1
V.sp.site <- kronecker(Vphy.site, Vphy.sp)
Y.sp.site.attract <- array(t(rmvnorm(n = 1, sigma = sd.sp.site.attract^2*V.sp.site)))

# Simulate residual error
sd.e <- 0.5
Y.e <- rnorm(nspp*nsite, sd = sd.e)

# Construct the dataset
d <- data.frame(sp = rep(phy.sp$tip.label, times = nsite),
                site = rep(phy.site$tip.label, each = nspp))

# Simulate abundance data
d$Y <- Y.sp + Y.site + Y.sp.attract + Y.site.attract + Y.sp.site.attract + Y.e

# Plot random effects covariance matrices and then add phylogenies
# Note that, if show.image and show.sim are not specified, pglmm_plot_ranef() shows
# the covariance matrices if nspp * nsite < 200 and shows simulations
# if nspp * nsite > 100
```

```
re <- pglmm_plot_ranef(Y ~ 1 + (1|sp__) + (1|site__) + (1|sp__@site) +
                    (1|sp@site__) + (1|sp__@site__),
                    data=d, cov_ranef = list(sp = phy.sp, site = phy.site))

# This flips the phylogeny to match to covariance matrices
rot.phy.site <- phy.site
for(i in (nsite+1):(nsite+Nnode(phy.site)))
   rot.phy.site <- rotate(rot.phy.site, node = i)

plot(phy.sp, main = "Species", direction = "upward")
plot(rot.phy.site, main = "Site")

# Analyze the simulated data and compute a P-value for the (1|sp__@site__)
# random effect using a LRT. It is often better to fit the reduced model before
# the full model, because it s numerically easier to fit the reduced model,
# and then the parameter estimates from the reduced model can be given to the
# full model. In this case, I have used the estimates of the random effects
# from the reduce model, mod.r$ss, as the initial estimates for the same
# parameters in the full model in the statement s2.init=c(mod.r$ss, 0.01)^2.
# The final 0.01 is for the last random effect in the full model, (1|sp__@site__).
# Note also that the output of the random effects from communityPGLMM(), mod.r$ss,
# are the standard deviations, so they have to be squared for use as initial
# values of variances in mod.f.

mod.r <- pglmm(Y ~ 1 + (1|sp__) + (1|site__) + (1|sp__@site) + (1|sp@site__),
                        data = d, cov_ranef = list(sp = phy.sp, site = phy.site))
mod.f <- pglmm(Y ~ 1 + (1|sp__) + (1|site__) + (1|sp__@site) + (1|sp@site__) +
               (1|sp__@site__), data = d,
               cov_ranef = list(sp = phy.sp, site = phy.site),
               s2.init = c(mod.r$ss, 0.01)^2)
mod.f
pvalue <- pchisq(2*(mod.f$logLik - mod.r$logLik), df = 1, lower.tail = FALSE)
pvalue
```

`pglmm_compare` *Phylogenetic Generalized Linear Mixed Model for Comparative Data*

Description

`pglmm_compare` performs linear regression for Gaussian, binomial and Poisson phylogenetic data, estimating regression coefficients with approximate standard errors. It simultaneously estimates the strength of phylogenetic signal in the residuals and gives an approximate conditional likelihood ratio test for the hypothesis that there is no signal. Therefore, when applied without predictor (independent) variables, it gives a test for phylogenetic signal. `pglmm_compare` is a wrapper for `pglmm` tailored for comparative data in which each value of the response (dependent) variable corresponds to a single tip on a phylogenetic tree. If there are multiple measures for each species, `pglmm` will be helpful.

Usage

```
pglmm_compare(
  formula,
  family = "gaussian",
  data = list(),
  phy,
  REML = TRUE,
  optimizer = c("nelder-mead-nlopt", "bobyqa", "Nelder-Mead", "subplex"),
  add.obs.re = TRUE,
  verbose = FALSE,
  cpp = TRUE,
  bayes = FALSE,
  reltol = 10^-6,
  maxit = 500,
  tol.pql = 10^-6,
  maxit.pql = 200,
  marginal.summ = "mean",
  calc.DIC = FALSE,
  prior = "inla.default",
  prior_alpha = 0.1,
  prior_mu = 1,
  ML.init = FALSE,
  s2.init = 1,
  B.init = NULL
)
```

Arguments

`formula`	A two-sided linear formula object describing the fixed-effects of the model; for example, Y ~ X. Binomial data can be either presence/absence, or a two-column array of 'successes' and 'failures'. For both binomial and Poisson data, we add an observation-level random term by default via `add.obs.re` = `TRUE`.
`family`	Either "gaussian" for a Linear Mixed Model, or "binomial" or "poisson" for Generalized Linear Mixed Models. `family` should be specified as a character string (i.e., quoted). For binomial and Poisson data, we use the canonical logit and log link functions, respectively. Binomial data can be either presence/absence, or a two-column array of 'successes' and 'failures'. For both Poisson and binomial data, we add an observation-level random term by default via `add.obs.re` = `TRUE`. If `bayes` = `TRUE` there are two additional families available: "zeroinflated.binomial", and "zeroinflated.poisson", which add a zero inflation parameter; this parameter gives the probability that the response is a zero. The rest of the parameters of the model then reflect the "non-zero" part of the model. Note that "zeroinflated.binomial" only makes sense for success/failure response data.
`data`	A data frame containing the variables named in formula. It must has the tip labels of the phylogeny as row names; if they are not in the same order, the data frame will be arranged so that row names match the order of tip labels.
`phy`	A phylogenetic tree as an object of class "phylo".

`REML`	Whether REML or ML is used for model fitting the random effects. Ignored if `bayes = TRUE`.
`optimizer`	nelder-mead-nlopt (default), bobyqa, Nelder-Mead, or subplex. Nelder-Mead is from the stats package and the other optimizers are from the nloptr package. Ignored if `bayes = TRUE`.
`add.obs.re`	Whether to add observation-level random term for binomial and Poisson families. Normally it would be a good idea to add this to account for overdispersion, so `add.obs.re = TRUE` by default.
`verbose`	If `TRUE`, the model deviance and running estimates of `s2` and `B` are plotted each iteration during optimization.
`cpp`	Whether to use C++ function for optim. Default is TRUE. Ignored if `bayes = TRUE`.
`bayes`	Whether to fit a Bayesian version of the PGLMM using `r-inla`. We recommend against Bayesian fitting for non-Gaussian data unless sample sizes are large (>1000), because the phylogenetic variance tends to get trapped near zero.
`reltol`	A control parameter dictating the relative tolerance for convergence in the optimization; see `optim`.
`maxit`	A control parameter dictating the maximum number of iterations in the optimization; see `optim`.
`tol.pql`	A control parameter dictating the tolerance for convergence in the PQL estimates of the mean components of the GLMM. Ignored if `family = "gaussian"` or `bayes = TRUE`.
`maxit.pql`	A control parameter dictating the maximum number of iterations in the PQL estimates of the mean components of the GLMM. Ignored if `family = "gaussian"` or `bayes = TRUE`.
`marginal.summ`	Summary statistic to use for the estimate of coefficients when doing a Bayesian PGLMM (when `bayes = TRUE`). Options are: "mean", "median", or "mode", referring to different characterizations of the central tendency of the Bayesian posterior marginal distributions. Ignored if `bayes = FALSE`.
`calc.DIC`	Should the Deviance Information Criterion be calculated and returned, when doing a Bayesian PGLMM? Ignored if `bayes = FALSE`.
`prior`	Which type of default prior should be used by `pglmm`? Only used if `bayes = TRUE`. There are currently four options: "inla.default", which uses the default `INLA` priors; "pc.prior.auto", which uses a complexity penalizing prior (as described in Simpson et al. (2017)) designed to automatically choose good parameters (only available for gaussian and binomial responses); "pc.prior", which allows the user to set custom parameters on the "pc.prior" prior, using the `prior_alpha` and `prior_mu` parameters (Run `INLA::inla.doc("pc.prec")` for details on these parameters); and "uninformative", which sets a very uninformative prior (nearly uniform) by using a very flat exponential distribution. The last option is generally not recommended but may in some cases give estimates closer to the maximum likelihood estimates. "pc.prior.auto" is only implemented for `family = "gaussian"` and `family = "binomial"` currently.
`prior_alpha`	Only used if `bayes = TRUE` and `prior = "pc.prior"`, in which case it sets the alpha parameter of `INLA`'s complexity penalizing prior for the random effects.The

	prior is an exponential distribution where prob(sd > mu) = alpha, where sd is the standard deviation of the random effect.
`prior_mu`	Only used if `bayes = TRUE` and `prior = "pc.prior"`, in which case it sets the mu parameter of `INLA`'s complexity penalizing prior for the random effects.The prior is an exponential distribution where prob(sd > mu) = alpha, where sd is the standard deviation of the random effect.
`ML.init`	Only relevant if `bayes = TRUE`. Should maximum likelihood estimates be calculated and used as initial values for the bayesian model fit? Sometimes this can be helpful, but most of the time it may not help; thus, we set the default to `FALSE`. Also, it does not work with the zero-inflated families.
`s2.init`	An array of initial estimates of s2. If s2.init is not provided for `family="gaussian"`, these are estimated using `lm` assuming no phylogenetic signal. If `s2.init` is not provided for `family = "binomial"`, these are set to 0.25.
`B.init`	Initial estimates of B, a matrix containing regression coefficients in the model for the fixed effects. This matrix must have `dim(B.init) = c(p + 1, 1)`, where `p` is the number of predictor (independent) variables; the first element of `B` corresponds to the intercept, and the remaining elements correspond in order to the predictor (independent) variables in the formula. If `B.init` is not provided, these are estimated using `lm` or `glm` assuming no phylogenetic signal.

Details

`pglmm_compare` in the package `phyr` is similar to `binaryPGLMM` in the package `ape`, although it has much broader functionality, including accepting more than just binary data, implementing Bayesian analyses, etc.

For non-Gaussian data, the function estimates parameters for the model

$$Pr(Y = 1) = \theta$$

$$\theta = inverse.link(b0 + b1 * x1 + b2 * x2 + \cdots + \epsilon)$$

$$\epsilon\ Gaussian(0, s2 * V)$$

where V is a covariance matrix derived from a phylogeny (typically under the assumption of Brownian motion evolution). Although mathematically there is no requirement for V to be ultrametric, forcing V into ultrametric form can aide in the interpretation of the model. This is especially true for binary data, because in regression for binary dependent variables, only the off-diagonal elements (i.e., covariances) of matrix V are biologically meaningful (see Ives & Garland 2014). The function converts a phylo tree object into a covariance matrix, and further standardizes this matrix to have determinant = 1. This in effect standardizes the interpretation of the scalar `s2`. Although mathematically not required, it is a very good idea to standardize the predictor (independent) variables to have mean 0 and variance 1. This will make the function more robust and improve the interpretation of the regression coefficients.

For Gaussian data, the function estimates parameters for the model

$$Y = b0 + b1 * x1 + b2 * x2 + \cdots + \epsilon)$$

$$\epsilon\ Gaussian(0, s2 * V + s2resid * I)$$

where $s2resid * I$ gives the non-phylogenetic residual variance. Note that this is equivalent to a model with Pagel's lambda transformation.

Value

An object (list) of class `pglmm_compare` with the following elements:

`formula`	the formula for fixed effects
`formula_original`	the formula for both fixed effects and random effects
`data`	the dataset
`family`	either `gaussian` or `binomial` or `poisson` depending on the model fit
`B`	estimates of the regression coefficients
`B.se`	approximate standard errors of the fixed effects regression coefficients. This is set to NULL if `bayes = TRUE`.
`B.ci`	approximate bayesian credible interval of the fixed effects regression coefficients. This is set to NULL if `bayes = FALSE`
`B.cov`	approximate covariance matrix for the fixed effects regression coefficients
`B.zscore`	approximate Z scores for the fixed effects regression coefficients. This is set to NULL if `bayes = TRUE`
`B.pvalue`	approximate tests for the fixed effects regression coefficients being different from zero. This is set to NULL if `bayes = TRUE`
`ss`	random effects' standard deviations for the covariance matrix $\sigma^2 V$ for each random effect in order. For the linear mixed model, the residual variance is listed last
`s2r`	random effects variances for non-nested random effects
`s2n`	random effects variances for nested random effects
`s2resid`	for linear mixed models, the residual variance
`s2r.ci`	Bayesian credible interval for random effects variances for non-nested random effects. This is set to NULL if `bayes = FALSE`
`s2n.ci`	Bayesian credible interval for random effects variances for nested random effects. This is set to NULL if `bayes = FALSE`
`s2resid.ci`	Bayesian credible interval for linear mixed models, the residual variance. This is set to NULL if `bayes = FALSE`
`logLik`	for linear mixed models, the log-likelihood for either the restricted likelihood (`REML=TRUE`) or the overall likelihood (`REML=FALSE`). This is set to NULL for generalised linear mixed models. If `bayes = TRUE`, this is the marginal log-likelihood
`AIC`	for linear mixed models, the AIC for either the restricted likelihood (`REML=TRUE`) or the overall likelihood (`REML=FALSE`). This is set to NULL for generalised linear mixed models
`BIC`	for linear mixed models, the BIC for either the restricted likelihood (`REML=TRUE`) or the overall likelihood (`REML=FALSE`). This is set to NULL for generalised linear mixed models

`DIC`	for bayesian PGLMM, this is the Deviance Information Criterion metric of model fit. This is set to NULL if `bayes = FALSE`.
`REML`	whether or not REML is used (`TRUE` or `FALSE`).
`bayes`	whether or not a Bayesian model was fit.
`marginal.summ`	The specified summary statistic used to summarise the Bayesian marginal distributions. Only present if `bayes = TRUE`
`s2.init`	the user-provided initial estimates of `s2`
`B.init`	the user-provided initial estimates of `B`
`Y`	the response (dependent) variable returned in matrix form
`X`	the predictor (independent) variables returned in matrix form (including 1s in the first column)
`H`	the residuals. For linear mixed models, this does not account for random terms, To get residuals after accounting for both fixed and random terms, use `residuals()`. For the generalized linear mixed model, these are the predicted residuals in the logit -1 space.
`iV`	the inverse of the covariance matrix. This is NULL if `bayes = TRUE`.
`mu`	predicted mean values for the generalized linear mixed model (i.e. similar to `fitted(merMod)`). Set to NULL for linear mixed models, for which we can use `fitted()`.
`Zt`	the design matrix for random effects. This is set to NULL if `bayes = TRUE`
`St`	diagonal matrix that maps the random effects variances onto the design matrix
`convcode`	the convergence code provided by `optim`. This is set to NULL if `bayes = TRUE`
`niter`	number of iterations performed by `optim`. This is set to NULL if `bayes = TRUE`
`inla.model`	Model object fit by underlying `inla` function. Only returned if `bayes = TRUE`

Author(s)

Anthony R. Ives

References

Ives, A. R. and Helmus, M. R. (2011) Generalized linear mixed models for phylogenetic analyses of community structure. *Ecological Monographs*, **81**, 511–525.

Ives, A. R. and Garland, T., Jr. (2014) Phylogenetic regression for binary dependent variables. Pages 231–261 *in* L. Z. Garamszegi, editor. *Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology*. Springer-Verlag, Berlin Heidelberg.

See Also

`pglmm`; package **ape** and its function `binaryPGLMM`; package **phylolm** and its function `phyloglm`; package **MCMCglmm**

Examples

```
## Illustration of `pglmm_compare` with simulated data

# Generate random phylogeny

library(ape)

n <- 100
phy <- compute.brlen(rtree(n=n), method = "Grafen", power = 1)

# Generate random data and standardize to have mean 0 and variance 1
X1 <- rTraitCont(phy, model = "BM", sigma = 1)
X1 <- (X1 - mean(X1))/var(X1)

# Simulate binary Y
sim.dat <- data.frame(Y = array(0, dim = n), X1 = X1, row.names = phy$tip.label)
sim.dat$Y <- ape::binaryPGLMM.sim(Y ~ X1, phy = phy, data=sim.dat, s2 = 1,
                             B = matrix(c(0, .25), nrow = 2, ncol = 1),
                             nrep = 1)$Y

# Fit model
pglmm_compare(Y ~ X1, family = "binomial", phy = phy, data = sim.dat)

# Compare with `binaryPGLMM`
ape::binaryPGLMM(Y ~ X1, phy = phy, data = sim.dat)

# Compare with `phyloglm`
summary(phylolm::phyloglm(Y ~ X1, phy = phy, data = sim.dat))

# Compare with `glm` that does not account for phylogeny
summary(glm(Y ~ X1, data = sim.dat, family = "binomial"))

# Compare with logistf() that does not account
# for phylogeny but is less biased than glm()
logistf::logistf(Y ~ X1, data = sim.dat)

## Fit model with bayes = TRUE
# pglmm_compare(Y ~ X1, family = "binomial", phy = phy, data = sim.dat,
#               bayes = TRUE, calc.DIC = TRUE)

# Compare with `MCMCglmm`

V <- vcv(phy)
V <- V/max(V)
detV <- exp(determinant(V)$modulus[1])
V <- V/detV^(1/n)

invV <- Matrix::Matrix(solve(V),sparse = TRUE)
sim.dat$species <- phy$tip.label
rownames(invV) <- sim.dat$species

nitt <- 43000
```

```
thin <- 10
burnin <- 3000

prior <- list(R=list(V=1, fix=1), G=list(G1=list(V=1, nu=1000, alpha.mu=0, alpha.V=1)))
# commented out to save time
# summary(MCMCglmm::MCMCglmm(Y ~ X1, random = ~species, ginvers = list(species = invV),
#     data = sim.dat, slice = TRUE, nitt = nitt, thin = thin, burnin = burnin,
#    family = "categorical", prior = prior, verbose = FALSE))
```

`pglmm_matrix_structure` — `pglmm_matrix_structure` *produces the entire covariance matrix structure (V) when you specify random effects.*

Description

`pglmm_matrix_structure` produces the entire covariance matrix structure (V) when you specify random effects.

Usage

```
pglmm_matrix_structure(
  formula,
  data = list(),
  family = "binomial",
  cov_ranef,
  repulsion = FALSE,
  ss = 1,
  cpp = TRUE
)

communityPGLMM.matrix.structure(
  formula,
  data = list(),
  family = "binomial",
  cov_ranef,
  repulsion = FALSE,
  ss = 1,
  cpp = TRUE
)
```

Arguments

`formula`	A two-sided linear formula object describing the mixed effects of the model. To specify that a random term should have phylogenetic covariance matrix along with non-phylogenetic one, add `__` (two underscores) at the end of the group

	variable; e.g., `+ (1 | sp__)` will construct two random terms, one with phylogenetic covariance matrix and another with non-phylogenetic (identity) matrix. In contrast, `__` in the nested terms (below) will only create a phylogenetic covariance matrix. Nested random terms have the general form `(1|sp__@site__)` which represents phylogenetically related species nested within correlated sites. This form can be used for bipartite questions. For example, species could be phylogenetically related pollinators and sites could be phylogenetically related plants, leading to the random effect `(1|insects__@plants__)`. If more than one phylogeny is used, remember to add all to the argument `cov_ranef = list(insects = insect_phylo, plants = plant_phylo)`. Phylogenetic correlations can be dropped by removing the `__` underscores. Thus, the form `(1|sp@site__)` excludes the phylogenetic correlations among species, while the form `(1|sp__@site)` excludes the correlations among sites. Note that correlated random terms are not allowed. For example, `(x|g)` will be the same as `(0 + x|g)` in the `lme4::lmer` syntax. However, `(x1 + x2|g)` won't work, so instead use `(x1|g) + (x2|g)`.
`data`	A `data.frame` containing the variables named in formula.
`family`	Either "gaussian" for a Linear Mixed Model, or "binomial" or "poisson" for Generalized Linear Mixed Models. "family" should be specified as a character string (i.e., quoted). For binomial and Poisson data, we use the canonical logit and log link functions, respectively. Binomial data can be either presence/absence, or a two-column array of 'successes' and 'failures'. For both binomial and Poisson data, we add an observation-level random term by default via `add.obs.re = TRUE`. If `bayes = TRUE` there are two additional families available: "zeroinflated.binomial", and "zeroinflated.poisson", which add a zero inflation parameter; this parameter gives the probability that the response is a zero. The rest of the parameters of the model then reflect the "non-zero" part part of the model. Note that "zeroinflated.binomial" only makes sense for success/failure response data.
`cov_ranef`	A named list of covariance matrices of random terms. The names should be the group variables that are used as random terms with specified covariance matrices (without the two underscores, e.g. `list(sp = tree1, site = tree2)`). The actual object can be either a phylogeny with class "phylo" or a prepared covariance matrix. If it is a phylogeny, `pglmm` will prune it and then convert it to a covariance matrix assuming Brownian motion evolution. `pglmm` will also standardize all covariance matrices to have determinant of one. Group variables will be converted to factors and all covariance matrices will be rearranged so that rows and columns are in the same order as the levels of their corresponding group variables.
`repulsion`	When there are nested random terms specified, `repulsion = FALSE` tests for phylogenetic underdispersion while `repulsion = FALSE` tests for overdispersion. This argument is a logical vector of length either 1 or >1. If its length is 1, then all covariance matrices in nested terms will be either inverted (overdispersion) or not. If its length is >1, then you can select which covariance matrix in the nested terms to be inverted. Make sure to get the length right: for all the terms with `@`, count the number of "__" to determine the length of repulsion. For example, `sp__@site` and `sp@site__` will each require one element

	of `repulsion`, while `sp__@site__` will take two elements (repulsion for sp and repulsion for site). Therefore, if your nested terms are `(1|sp__@site) + (1|sp@site__) + (1|sp__@site__)`, then you should set the repulsion to be something like `c(TRUE, FALSE, TRUE, TRUE)` (length of 4).
`ss`	Which of the `random.effects` to produce.
`cpp`	Whether to use C++ function for optim. Default is TRUE. Ignored if `bayes = TRUE`.

Value

A design matrix.

`pglmm_plot_ranef` *Visualize random terms of communityPGLMMs*

Description

Plot variance-cov matrix of random terms; also it is optional to simulate and visualize data based on these var-cov matrices. The input can be a communityPGLMM model (by setting argument `x`). If no model has been fitted, you can also specify data, formula, and family, etc. without actually fitting the model, which will save time.

Usage

```
pglmm_plot_ranef(
  formula = NULL,
  data = NULL,
  family = "gaussian",
  sp.var = "sp",
  site.var = "site",
  tree = NULL,
  tree_site = NULL,
  repulsion = FALSE,
  x = NULL,
  show.image = TRUE,
  show.sim.image = FALSE,
  random.effects = NULL,
  add.tree.sp = TRUE,
  add.tree.site = FALSE,
  cov_ranef = NULL,
  tree.panel.space = 0.5,
  title.space = 5,
  tree.size = 3,
  ...
)
```

```
communityPGLMM.show.re(
  formula = NULL,
  data = NULL,
  family = "gaussian",
  sp.var = "sp",
  site.var = "site",
  tree = NULL,
  tree_site = NULL,
  repulsion = FALSE,
  x = NULL,
  show.image = TRUE,
  show.sim.image = FALSE,
  random.effects = NULL,
  add.tree.sp = TRUE,
  add.tree.site = FALSE,
  cov_ranef = NULL,
  tree.panel.space = 0.5,
  title.space = 5,
  tree.size = 3,
  ...
)

pglmm_plot_re(
  formula = NULL,
  data = NULL,
  family = "gaussian",
  sp.var = "sp",
  site.var = "site",
  tree = NULL,
  tree_site = NULL,
  repulsion = FALSE,
  x = NULL,
  show.image = TRUE,
  show.sim.image = FALSE,
  random.effects = NULL,
  add.tree.sp = TRUE,
  add.tree.site = FALSE,
  cov_ranef = NULL,
  tree.panel.space = 0.5,
  title.space = 5,
  tree.size = 3,
  ...
)

communityPGLMM.plot.re(
  formula = NULL,
  data = NULL,
  family = "gaussian",
```

```
  sp.var = "sp",
  site.var = "site",
  tree = NULL,
  tree_site = NULL,
  repulsion = FALSE,
  x = NULL,
  show.image = TRUE,
  show.sim.image = FALSE,
  random.effects = NULL,
  add.tree.sp = TRUE,
  add.tree.site = FALSE,
  cov_ranef = NULL,
  tree.panel.space = 0.5,
  title.space = 5,
  tree.size = 3,
  ...
)
```

Arguments

`formula`	A two-sided linear formula object describing the mixed effects of the model. To specify that a random term should have phylogenetic covariance matrix along with non-phylogenetic one, add `__` (two underscores) at the end of the group variable; e.g., `+ (1 | sp__)` will construct two random terms, one with phylogenetic covariance matrix and another with non-phylogenetic (identity) matrix. In contrast, `__` in the nested terms (below) will only create a phylogenetic covariance matrix. Nested random terms have the general form `(1|sp__@site__)` which represents phylogenetically related species nested within correlated sites. This form can be used for bipartite questions. For example, species could be phylogenetically related pollinators and sites could be phylogenetically related plants, leading to the random effect `(1|insects__@plants__)`. If more than one phylogeny is used, remember to add all to the argument `cov_ranef = list(insects = insect_phylo, plants = plant_phylo)`. Phylogenetic correlations can be dropped by removing the `__` underscores. Thus, the form `(1|sp@site__)` excludes the phylogenetic correlations among species, while the form `(1|sp__@site)` excludes the correlations among sites. Note that correlated random terms are not allowed. For example, `(x|g)` will be the same as `(0 + x|g)` in the `lme4::lmer` syntax. However, `(x1 + x2|g)` won't work, so instead use `(x1|g) + (x2|g)`.
`data`	A `data.frame` containing the variables named in formula.
`family`	Either "gaussian" for a Linear Mixed Model, or "binomial" or "poisson" for Generalized Linear Mixed Models. "family" should be specified as a character string (i.e., quoted). For binomial and Poisson data, we use the canonical logit and log link functions, respectively. Binomial data can be either presence/absence, or a two-column array of 'successes' and 'failures'. For both binomial and Poisson data, we add an observation-level random term by default via `add.obs.re = TRUE`. If `bayes = TRUE` there are two additional families available: "zeroinflated.binomial", and "zeroinflated.poisson", which add a zero inflation param-

	eter; this parameter gives the probability that the response is a zero. The rest of the parameters of the model then reflect the "non-zero" part part of the model. Note that "zeroinflated.binomial" only makes sense for success/failure response data.
`sp.var`	The variable name of "species"; y-axis of the image.
`site.var`	The variable name of "site"; x-axis of the image.
`tree`	A phylogeny for column sp, with "phylo" class, or a covariance matrix for sp. Make sure to have all species in the matrix; if the matrix is not standardized, (i.e., det(tree) != 1), `pglmm` will try to standardize it for you. No longer used: keep here for compatibility.
`tree_site`	A second phylogeny for "site". This is required only if the site column contains species instead of sites. This can be used for bipartitie questions; tree_site can also be a covariance matrix. Make sure to have all sites in the matrix; if the matrix is not standardized (i.e., det(tree_site) != 1), pglmm‘ will try to standardize it for you. No longer used: keep here for compatibility.
`repulsion`	When there are nested random terms specified, `repulsion = FALSE` tests for phylogenetic underdispersion while `repulsion = FALSE` tests for overdispersion. This argument is a logical vector of length either 1 or >1. If its length is 1, then all covariance matrices in nested terms will be either inverted (overdispersion) or not. If its length is >1, then you can select which covariance matrix in the nested terms to be inverted. Make sure to get the length right: for all the terms with `@`, count the number of "__" to determine the length of repulsion. For example, `sp__@site` and `sp@site__` will each require one element of `repulsion`, while `sp__@site__` will take two elements (repulsion for sp and repulsion for site). Therefore, if your nested terms are `(1|sp__@site) + (1|sp@site__) + (1|sp__@site__)`, then you should set the repulsion to be something like `c(TRUE, FALSE, TRUE, TRUE)` (length of 4).
`x`	A fitted model with class communityPGLMM.
`show.image`	Whether to show the images of random effects.
`show.sim.image`	Whether to show the images of simulated site by sp matrix. This can be useful to see how the phylogenetic information were included.
`random.effects`	Optional pre-build list of random effects. If `NULL` (the default), the function `prep_dat_pglmm` will prepare the random effects for you from the information in `formula`, `data`, and `cov_ranef`. `random.effect` allows a list of pre-generated random effects terms to increase flexibility; for example, this makes it possible to construct models with both phylogenetic correlation and spatio-temporal autocorrelation. In preparing `random.effect`, make sure that the orders of rows and columns of covariance matrices in the list are the same as their corresponding group variables in the data. Also, this should be *a list of lists*, e.g. `random.effects = list(re1 = list(matrix_a), re2 = list(1, sp = sp, covar = Vsp))`.
`add.tree.sp`	Whether to add a phylogeny of species at the top of the simulated site by sp matrix plot, default is TRUE.
`add.tree.site`	Whether to add a phylogeny of sites at the right of the simulated site by sp matrix plot, default is FALSE.

`cov_ranef`	A named list of covariance matrices of random terms. The names should be the group variables that are used as random terms with specified covariance matrices (without the two underscores, e.g. `list(sp = tree1, site = tree2)`). The actual object can be either a phylogeny with class "phylo" or a prepared covariance matrix. If it is a phylogeny, `pglmm` will prune it and then convert it to a covariance matrix assuming Brownian motion evolution. `pglmm` will also standardize all covariance matrices to have determinant of one. Group variables will be converted to factors and all covariance matrices will be rearranged so that rows and columns are in the same order as the levels of their corresponding group variables.
`tree.panel.space`	The number of lines between the phylogeny and the matrix plot, if add.tree is TRUE.
`title.space`	The number of lines between the title and the matrix plot, if add.tree is TRUE.
`tree.size`	The height of the phylogeny to be plotted (number of lines), if add.tree is TRUE.
`...`	Additional arguments for `Matrix::image()` or `lattice::levelplot()`. Common ones are: • `useAbs` whether to use absolute values of the matrix; if no negative values, this will be set to TRUE if not specified. When `useAbs = TRUE` the color scheme will be black-white, otherwise, it will be red/blue. • `colorkey` whether to draw the scale legend at the right side of each plot?

Value

A hidden list, including the covariance matrices and simulated site by species matrices. Individual plots are saved as `plt_re_list` and `plt_sim_list`. If `show.image` or `show.sim.image` is TRUE, the corresponding final plot (`plt_re_all_in_one` or `plt_sim_all_in_one`) can be saved as external file using `ggplot2::ggsave` as it is a grid object.

`pglmm_predicted_values`

Predicted values of PGLMM

Description

`pglmm_predicted_values` calculates the predicted values of Y; for the generalized linear mixed model (family %in% c("binomial","poisson"), these values are in the transformed space.

Usage

```
pglmm_predicted_values(
  x,
  cpp = TRUE,
  gaussian.pred = c("nearest_node", "tip_rm"),
  re.form = NULL,
```

```
  type = c("link", "response"),
  ...
)

communityPGLMM.predicted.values(
  x,
  cpp = TRUE,
  gaussian.pred = c("nearest_node", "tip_rm")
)
```

Arguments

`x`	A fitted model with class communityPGLMM.
`cpp`	Whether to use c++ code. Default is TRUE.
`gaussian.pred`	When family is gaussian, which type of prediction to calculate? Option nearest_node will predict values to the nearest node, which is same as lme4::predict or fitted. Option tip_rm will remove the point then predict the value of this point with remaining ones.
`re.form`	(formula, `NULL`, or `NA`) specify which random effects to condition on when predicting. If `NULL`, include all random effects (i.e Xb + Zu); if `NA` or `~0`, include no random effects (i.e. Xb).
`type`	character string - either `"link"`, the default, or `"response"` indicating the type of prediction object returned.
`...`	Optional additional parameters. None are used at present.

Value

A data frame with column Y_hat (predicted values accounting for both fixed and random terms).

`pglmm_profile_LRT` `pglmm_profile_LRT` *tests statistical significance of the phylogenetic random effect of binomial models on species slopes using a likelihood ratio test.*

Description

`pglmm_profile_LRT` tests statistical significance of the phylogenetic random effect of binomial models on species slopes using a likelihood ratio test.

Usage

```
pglmm_profile_LRT(x, re.number = 0, cpp = TRUE)

communityPGLMM.profile.LRT(x, re.number = 0, cpp = TRUE)
```

Arguments

`x`	A fitted model with class communityPGLMM and family "binomial".
`re.number`	Which random term to test? Can be a vector with length >1
`cpp`	Whether to use C++ function for optim. Default is TRUE. Ignored if `bayes = TRUE`.

Value

A list of likelihood, df, and p-value.

`phylotree` *Example phylogeny*

Description

A phylogeny with more species than the community data.

Usage

```
phylotree
```

Format

Newick format.

`plot_bayes` *plot_bayes generic*

Description

plot_bayes generic

Usage

```
plot_bayes(x, ...)
```

Arguments

`x`	A communityPGLMM object fit with `bayes = TRUE`.
`...`	Further arguments to pass to or from other methods.

Value

A ggplot object

---

plot_data *Plot the original dataset and predicted values (optional)*

Description

Plots a representation of the marginal posterior distribution of model parameters. Note this function requires the packages `ggplot2` and `ggridges` to be installed.

Usage

```
plot_data(
  x,
  sp.var = "sp",
  site.var = "site",
  show.sp.names = FALSE,
  show.site.names = FALSE,
  digits = max(3, getOption("digits") - 3),
  predicted = FALSE,
  ...
)

## S3 method for class 'communityPGLMM'
plot_bayes(x, n_samp = 1000, sort = TRUE, ...)
```

Arguments

`x`	A communityPGLMM object fit with `bayes = TRUE`.
`sp.var`	The variable name of "species"; y-axis of the image.
`site.var`	The variable name of "site"; x-axis of the image.
`show.sp.names`	Whether to print species names as y-axis labels.
`show.site.names`	Whether to print site names as x-axis labels.
`digits`	Not used.
`predicted`	Whether to plot predicted values side by side with observed ones.
`...`	Further arguments to pass to or from other methods.
`n_samp`	Number of sample from the marginal posterior to take in order to estimate the posterior density.
`sort`	Whether to plot different terms in the order of their estimations. Default is 'TRUE'.

Value

A ggplot object

Note

The underlying plot grid object is returned but invisible. It can be saved for later uses.

`predict.communityPGLMM` *Predict Function for communityPGLMM Model Objects*

Description

Predict Function for communityPGLMM Model Objects

Usage

```
## S3 method for class 'communityPGLMM'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	Object of class inheriting from `"lm"`
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`...`	further arguments passed to or from other methods.

Value

The form of the value returned by `predict` depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

`prep_dat_pglmm` *Prepare data for* `pglmm`

Description

This function is mainly used within `pglmm` but can also be used independently to prepare a list of random effects, which then can be updated by users for more complex models.

Usage

```
prep_dat_pglmm(
  formula,
  data,
  cov_ranef = NULL,
  repulsion = FALSE,
  prep.re.effects = TRUE,
  family = "gaussian",
  add.obs.re = TRUE,
  bayes = FALSE,
  bayes_nested_matrix_as_list = FALSE
)
```

Arguments

`formula`	A two-sided linear formula object describing the mixed effects of the model. To specify that a random term should have phylogenetic covariance matrix along with non-phylogenetic one, add `__` (two underscores) at the end of the group variable; e.g., `+ (1 | sp__)` will construct two random terms, one with phylogenetic covariance matrix and another with non-phylogenetic (identity) matrix. In contrast, `__` in the nested terms (below) will only create a phylogenetic covariance matrix. Nested random terms have the general form `(1|sp__@site__)` which represents phylogenetically related species nested within correlated sites. This form can be used for bipartite questions. For example, species could be phylogenetically related pollinators and sites could be phylogenetically related plants, leading to the random effect `(1|insects__@plants__)`. If more than one phylogeny is used, remember to add all to the argument `cov_ranef = list(insects = insect_phylo, plants = plant_phylo)`. Phylogenetic correlations can be dropped by removing the `__` underscores. Thus, the form `(1|sp@site__)` excludes the phylogenetic correlations among species, while the form `(1|sp__@site)` excludes the correlations among sites. Note that correlated random terms are not allowed. For example, `(x|g)` will be the same as `(0 + x|g)` in the `lme4::lmer` syntax. However, `(x1 + x2|g)` won't work, so instead use `(x1|g) + (x2|g)`.
`data`	A `data.frame` containing the variables named in formula.
`cov_ranef`	A named list of covariance matrices of random terms. The names should be the group variables that are used as random terms with specified covariance matrices (without the two underscores, e.g. `list(sp = tree1, site = tree2)`). The actual object can be either a phylogeny with class "phylo" or a prepared covariance matrix. If it is a phylogeny, `pglmm` will prune it and then convert it to a covariance matrix assuming Brownian motion evolution. `pglmm` will also standardize all covariance matrices to have determinant of one. Group variables will be converted to factors and all covariance matrices will be rearranged so that rows and columns are in the same order as the levels of their corresponding group variables.
`repulsion`	When there are nested random terms specified, `repulsion = FALSE` tests for phylogenetic underdispersion while `repulsion = FALSE` tests for overdisper-

	sion. This argument is a logical vector of length either 1 or >1. If its length is 1, then all covariance matrices in nested terms will be either inverted (overdispersion) or not. If its length is >1, then you can select which covariance matrix in the nested terms to be inverted. Make sure to get the length right: for all the terms with `@`, count the number of "__" to determine the length of repulsion. For example, `sp__@site` and `sp@site__` will each require one element of `repulsion`, while `sp__@site__` will take two elements (repulsion for sp and repulsion for site). Therefore, if your nested terms are `(1|sp__@site) + (1|sp@site__) + (1|sp__@site__)`, then you should set the repulsion to be something like `c(TRUE, FALSE, TRUE, TRUE)` (length of 4).
`prep.re.effects`	Whether to prepare random effects for users.
`family`	Either "gaussian" for a Linear Mixed Model, or "binomial" or "poisson" for Generalized Linear Mixed Models. "family" should be specified as a character string (i.e., quoted). For binomial and Poisson data, we use the canonical logit and log link functions, respectively. Binomial data can be either presence/absence, or a two-column array of 'successes' and 'failures'. For both binomial and Poisson data, we add an observation-level random term by default via `add.obs.re = TRUE`. If `bayes = TRUE` there are two additional families available: "zeroinflated.binomial", and "zeroinflated.poisson", which add a zero inflation parameter; this parameter gives the probability that the response is a zero. The rest of the parameters of the model then reflect the "non-zero" part part of the model. Note that "zeroinflated.binomial" only makes sense for success/failure response data.
`add.obs.re`	Whether to add an observation-level random term for binomial or Poisson distributions. Normally it would be a good idea to add this to account for overdispersion, so `add.obs.re = TRUE` by default.
`bayes`	Whether to fit a Bayesian version of the PGLMM using `r-inla`.
`bayes_nested_matrix_as_list`	For `bayes = TRUE`, prepare the nested terms as a list of length of 4 as the old way?

Value

A list with updated formula, random.effects, and updated cov_ranef.

`print.communityPGLMM` *Print summary information of fitted model*

Description

Print summary information of fitted model

Usage

```
## S3 method for class 'communityPGLMM'
print(x, digits = max(3, getOption(”digits”) - 3), ...)
```

Arguments

`x`	A fitted communityPGLMM model.
`digits`	Minimal number of significant digits for printing, as in `print.default`.
`...`	Additional arguments, currently ignored.

`print.pglmm_compare` *Print summary information of fitted model*

Description

Print summary information of fitted model

Usage

```
## S3 method for class 'pglmm_compare'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	A fitted pglmm_compare.
`digits`	Minimal number of significant digits for printing, as in `print.default`.
`...`	Additional arguments, currently ignored.

`psv` *Phylogenetic Species Diversity Metrics*

Description

Calculate the bounded phylogenetic biodiversity metrics: phylogenetic species variability, richness, evenness and clustering for one or multiple communities.

Usage

```
psv(
  comm,
  tree,
  compute.var = TRUE,
  scale.vcv = TRUE,
  prune.tree = FALSE,
  cpp = TRUE
)

psr(
  comm,
```

```
  tree,
  compute.var = TRUE,
  scale.vcv = TRUE,
  prune.tree = FALSE,
  cpp = TRUE
)

pse(comm, tree, scale.vcv = TRUE, prune.tree = FALSE, cpp = TRUE)

psc(comm, tree, scale.vcv = TRUE, prune.tree = FALSE)

psv.spp(comm, tree, scale.vcv = TRUE, prune.tree = FALSE, cpp = TRUE)

psd(
  comm,
  tree,
  compute.var = TRUE,
  scale.vcv = TRUE,
  prune.tree = FALSE,
  cpp = TRUE
)
```

Arguments

`comm`	Community data matrix, site as rows and species as columns, site names as row names.
`tree`	A phylo tree object with class "phylo" or a phylogenetic covariance matrix.
`compute.var`	Logical, default is TRUE, computes the expected variances for PSV and PSR for each community.
`scale.vcv`	Logical, default is TRUE, scale the phylogenetic covariance matrix to bound the metric between 0 and 1 (i.e. correlations).
`prune.tree`	Logical, default is FALSE, prune the phylogeny before converting to var-cov matrix? Pruning and then converting VS converting then subsetting may have different var-cov matrix resulted.
`cpp`	Logical, default is TRUE, whether to use cpp for internal calculations.

Details

Phylogenetic species variability (PSV) quantifies how phylogenetic relatedness decreases the variance of a hypothetical unselected/neutral trait shared by all species in a community. The expected value of PSV is statistically independent of species richness, is one when all species in a community are unrelated (i.e., a star phylogeny) and approaches zero as species become more related. PSV is directly related to mean phylogenetic distance, except except calculated on a scaled phylogenetic covariance matrix. The expected variance around PSV for any community of a particular species richness can be approximated. To address how individual species contribute to the mean PSV of a data set, the function `psv.spp` gives signed proportions of the total deviation from the mean PSV that occurs when all species are removed from the data set one at a time. The absolute values of these “species effects” tend to positively correlate with species prevalence.

Value

Returns a dataframe of the respective phylogenetic species diversity metric values

Note

These metrics are bounded either between zero and one (PSV, PSE, PSC) or zero and species richness (PSR); but the metrics asymptotically approach zero as relatedness increases. Zero can be assigned to communities with less than two species, but conclusions drawn from assigning communities zero values need be carefully explored for any data set. The data sets need not be species-community data sets but may be any community data set with an associated phylogeny.

Author(s)

Matthew Helmus `<mrhelmus@gmail.com>`

References

Helmus M.R., Bland T.J., Williams C.K. & Ives A.R. 2007. Phylogenetic measures of biodiversity. American Naturalist, 169, E68-E83

Examples

```
psv(comm = comm_a, tree = phylotree)
```

`ranef` *Extract random-effects estimates*

Description

Extract the random-effects estimates

Usage

```
## S3 method for class 'communityPGLMM'
ranef(object, ...)
```

Arguments

`object`	A fitted model with class communityPGLMM.
`...`	Ignored.

Details

Extract the estimates of the random-effects parameters from a fitted model.

Value

A dataframe of random-effects estimates.

`refit_boots` *Refit bootstrap replicates that failed to converge in a call to* `cor_phylo`

Description

This function is to be called on a `cor_phylo` object if when one or more bootstrap replicates fail to converge. It allows the user to change parameters for the optimizer to get it to converge. One or more of the resulting `cp_refits` object(s) can be supplied to `boot_ci` along with the original `cor_phylo` object to calculate confidence intervals from only bootstrap replicates that converged.

Usage

```
refit_boots(cp_obj, inds = NULL, ...)

## S3 method for class 'cp_refits'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`cp_obj`	The original `cor_phylo` object that was bootstrapped.
`inds`	Vector of indices indicating the bootstraps you want to refit. This is useful if you want to try refitting only a portion of bootstrap replicates. By passing `NULL`, it refits all bootstrap replicates present in `cp_obj$bootstrap$mats`. Any bootstrap replicates not present in `inds` will have `NA` in the output object. Defaults to `NULL`.
`...`	Arguments that should be changed from the original call to `cor_phylo`. The `boot` argument is always set to `0` for refits because you don't want to bootstrap your bootstraps.
`x`	an object of class `cp_refits`.
`digits`	the number of digits to be printed.

Value

A `cp_refits` object, which is a list of `cor_phylo` objects corresponding to each matrix in `<original cor_phylo object>$b`

Methods (by generic)

- `print`: prints `cp_refits` objects

`residuals.communityPGLMM` *Residuals of communityPGLMM objects*

Description

Getting different types of residuals for communityPGLMM objects.

Usage

```
## S3 method for class 'communityPGLMM'
residuals(
  object,
 type = if (object$family %in% c("binomial", "poisson")) "deviance" else "response",
  scaled = FALSE,
  ...
)
```

Arguments

`object`	A fitted model with class communityPGLMM.
`type`	Type of residuals, currently only "response" for gaussian pglmm; "deviance" (default) and "response" for binomial and poisson pglmm.
`scaled`	Scale residuals by residual standard deviation for gaussian pglmm.
`...`	Additional arguments, ignored for method compatibility.

Value

A vector of residuals.

`rm_site_noobs` *Remove site that has no observations of any species*

Description

This function will remove site that has no observations in a site by species data frame.

Usage

```
rm_site_noobs(df, warn = FALSE)
```

Arguments

`df`	A data frame in wide form, i.e. site by species data frame, with site names as row name.
`warn`	Whether to warn when any site has no species? Default is `FALSE`.

Value

A site by species data frame.

Author(s)

Daijiang Li

`rm_sp_noobs` *Remove species that not observed in any site*

Description

Remove species that not observed in any site

Usage

```
rm_sp_noobs(df, warn = FALSE)
```

Arguments

`df`	A data frame in wide form, i.e. site by species data frame, with site names as row name.
`warn`	Whether to warn when any species does not occur in at least one site? Default is `FALSE`.

Value

A site by species data frame.

Author(s)

Daijiang Li

This function will remove species that has no observations in any site.

---

simulate.communityPGLMM

Simulate from a communityPGLMM object

Description

Note that this function currently only works for model fit with `bayes = TRUE`

Usage

```
## S3 method for class 'communityPGLMM'
simulate(object, nsim = 1, seed = NULL, re.form = NULL, ...)
```

Arguments

`object`	A fitted model object with class 'communityPGLMM'.
`nsim`	positive integer scalar - the number of responses to simulate.
`seed`	an optional seed to be used in `set.seed` immediately before the simulation so as to generate a reproducible sample.
`re.form`	(formula, `NULL`, or `NA`) specify which random effects to condition on when predicting. If `NULL`, include all random effects and the conditional modes of those random effects will be included in the deterministic part of the simulation (i.e Xb + Zu); if `NA` or `~0`, include no random effects and new values will be chosen for each group based on the estimated random-effects variances (i.e. Xb + Zu * u_random).
`...`	optional additional arguments: none are used at present.

---

summary.communityPGLMM

Summary information of fitted model

Description

Summary information of fitted model

Usage

```
## S3 method for class 'communityPGLMM'
summary(object, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	A fitted model with class communityPGLMM.
`digits`	Minimal number of significant digits for printing, as in `print.default`.
`...`	Additional arguments, currently ignored.

`summary.pglmm_compare` *Summary information of fitted pglmm_compare model*

Description

Summary information of fitted pglmm_compare model

Usage

```
## S3 method for class 'pglmm_compare'
summary(object, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	A fitted model with class pglmm_compare.
`digits`	Minimal number of significant digits for printing, as in `print.default`.
`...`	Additional arguments, currently ignored.

`traits` *Example species traits data*

Description

A data frame of species functional traits.

Usage

```
traits
```

Format

A data frame with 18 species and 3 variables: sla, height, and seed dispersal mode.

`vcv2` *Create phylogenetic var-cov matrix*

Description

This function will convert a phylogeny to a Var-cov matrix.

Usage

```
vcv2(phy, corr = FALSE)
```

Arguments

`phy`	A phylogeny with "phylo" as class.
`corr`	Whether to return a correlation matrix instead of Var-cov matrix. Default is FALSE.

Value

A phylogenetic var-cov matrix.

`%nin%` *Not in*

Description

This function will return elements of x not in y

Usage

```
x %nin% y
```

Arguments

`x`	A vector.
`y`	A vector.

Value

A vector.

Index