Package ‘mvSLOUCH’

July 24, 2025

Type Package

Title Multivariate Stochastic Linear Ornstein-Uhlenbeck Models for Phylogenetic Comparative Hypotheses

Version 2.7.7

Date 2025-07-24

Maintainer Krzysztof Bartoszek <krzbar@protonmail.ch>

Description Fits multivariate Ornstein-Uhlenbeck types of models to continues trait data from species related by a common evolutionary history. See K. Bartoszek, J, Pienaar, P. Mostad, S. Andersson, T. F. Hansen (2012) <doi:10.1016/j.jtbi.2012.08.005> and K. Bartoszek, and J. Tredgett Clarke, J. Fuentes-Gonzalez, V. Mitov, J. Pienaar, M. Piwczynski, R. Puchalka, K. Spalik, K. L. Voje (2024) <doi:10.1111/2041-210X.14376>. The suggested PCMBaseCpp package (which significantly speeds up the likelihood calculations) can be obtained from <https://github.com/venelin/PCMBaseCpp/>.

Depends R(>= 3.5.0), abind

License GPL (>= 2) | file LICENCE

LazyLoad yes

Collate bootstrap.R evolmodelest.R matrixexps.R phylgls.R sdecovariancephyl.R wrappers.R ci.R fitch.mvsl.R calls_to_matrixcalc.R matrixparametrizations.R PhyloSDEestim.R sdemoments.R estimBM.R getESS.R modelparams.R phyltree_paths.R simulVasicekprocphyl.R estimGLSGC.R loglik.R modelparamssummary.R precalcs.R simulVasicekproc.R estimMAXLIK.R make.states.mvsl.R modelparamstransform.R regimes.R trees2slouch.mvsl.R simulclustphyl.R OUphylregression.R sdemeanphyl.R

Imports ape (>= 5.3), graphics, methods, mvtnorm, Matrix, ouch, PCMBase (>= 1.2.10), stats, matrixcalc

Suggests PCMBaseCpp (>= 0.1.9), ggplot2, knitr, rmarkdown, mvMORPH, testthat

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Krzysztof Bartoszek [cre, aut],
John Clarke [ctb],
Jesualdo Fuentes-Gonzalez [ctb],
Jason Pienaar [ctb]

Repository CRAN

Date/Publication 2025-07-24 14:40:02 UTC

Contents

`mvSLOUCH-package` *Multivariate Ornstein-Uhlenbeck type stochastic differential equation models for phylogenetic comparative data.*

Description

The package allows for maximum likelihood estimation, simulation and study of properties of multivariate Brownian motion

$$dX(t) = \Sigma dB(t),$$

OU

$$dY(t) = -A(Y(t) - \Psi(t))dt + \Sigma dB(t)$$

and OUBM

$$\begin{aligned} dY(t) &= -A(Y(t) - (\Psi(t) - A^{-1}BX(t)))dt + \Sigma_{yy}dB(t) \\ dX(t) &= \Sigma_{xx}dB(t) \end{aligned}$$

models that evolve on a phylogenetic tree.

This software comes AS IS in the hope that it will be useful WITHOUT ANY WARRANTY, NOT even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. Please understand that there may still be bugs and errors. Use it at your own risk. We take no responsibility for any errors or omissions in this package or for any misfortune that may befall you or others as a result of its use. Please send comments and report bugs to Krzysztof Bartoszek at krzbar@protonmail.ch .

Details

Package:	mvSLOUCH
Type:	Package
Version:	2.7.7
Date:	2025-07-24
License:	GPL (>= 2)
LazyLoad:	yes

The package allows for maximum likelihood estimation, simulation and study of properties of multivariate Brownian motion

$$dX(t) = \Sigma dB(t),$$

OU

$$dY(t) = -A(Y(t) - \Psi(t))dt + \Sigma dB(t)$$

and OUBM

$$\begin{aligned} dY(t) &= -A(Y(t) - \Psi(t) - A^{-1}BX(t))dt + \Sigma_{yy}dB(t) \\ dX(t) &= \Sigma_{xx}dB(t) \end{aligned}$$

models that evolve on a phylogenetic tree.

The estimation functions are `BrownianMotionModel`, `ouchModel` (OUOU) and `mvslouchModel` (mvOUBM). They rely on a combination of least squares and numerical optimization techniques. A wrapper function for all of them is `estimate.evolutionary.model`, it tries all three models with different matrix parameter classes and then returns the best model based on the AICc.

The simulation functions are simulBMProcPhylTree, simulOUCHProcPhylTree, simulMVSLOUCHProcPhylTree.

The phylogeny provided to them should be of the `phylo` (package **ape**) format.

The package uses the functions `.sym.par()` and `.sym.unpar()` from the **ouch** package to parametrize symmetric matrices.

In the case the mvOUBM model with a single response trait the package **slouch** is a recommended alternative.

The package uses **PCMBase**'s `PCMLik()` function as the engine to do calculate the likelihood and phylogenetic least squares. If the **PCMBaseCpp** package is installed **mvSLOUCH** can take advantage of it to significantly decrease the running time. The **PCMBaseCpp** package is available from `https://github.com/venelin/PCMBaseCpp`.

Author(s)

Krzysztof Bartoszek Maintainer: <krzbar@protonmail.ch>

References

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Felsenstein, J. (1985) Phylogenies and the comparative method. American Naturalist 125:1-15.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Hansen, T.F. and Pienaar, J. and Orzack, S.H. (2008) A comparative method for studying adaptation to randomly evolving environment. Evolution 62:1965-1977.

Labra, A., Pienaar, J. & Hansen, T.F. (2009) Evolution of thermophysiology in Liolaemus lizards: adaptation, phylogenetic inertia and niche tracking. The American Naturalist 174:204-220.

Mitov, V. and Bartoszek, K. and Asimomitis, G. and Stadler, T. (2020) Fast likelihood calculation for multivariate Gaussian phylogenetic models with shifts Theoretical Population Biology 131:66-78.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

Examples

```
## Set the version of the random number generator
## so that the results are always reproducible
RNGversion(min(as.character(getRversion()),"3.6.1"))
## Set the random seed, and random number generator
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
```

```
### For simulating the tree one could also use alternative functions,
## e.g. sim.bd.taxa from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed.
## It enhances the phylo object with multiple additional fields, e.g.,
## the lineage for each tip, that are used by subsequent mvSLOUCH functions.
phyltree<-phyltree_paths(phyltree)

## Define a vector of regimes. There are two of them: small and large;
## their layout is random.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the different models.
### There is no particular meaning attached to these parameters.
### BM parameters
BMparameters<-list(vX0=matrix(0,nrow=3,ncol=1),
Sxx=rbind(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))
### OUOU parameters
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
"large"=c(-1,1,0.5)),Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
### OUBM parameters
OUBMparameters<-list(vY0=matrix(c(1,-1),ncol=1,nrow=2),A=rbind(c(9,0),c(0,5)),
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data under 3D BM, OUOU and OUBM (the third variable is the
### BM one) models of evolution.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx)
### extract just the values at the tips of the phylogeny
BMdata<-BMdata[phyltree$tip.label,,drop=FALSE]
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
### extract just the values at the tips of the phylogeny
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
### extract just the values at the tips of the phylogeny
OUBMdata<-OUBMdata[phyltree$tip.label,,drop=FALSE]

### Recover the parameters of the SDEs.
### Recover under the BM model.
BMestim<-BrownianMotionModel(phyltree,BMdata)
### reset to the original version of the random number generator.
RNGversion(as.character(getRversion()))
## Not run:
### It takes too long to run this from this point.
### We do not reduce the number of iterations of the optimier
### (as we did in other manual page examples, so the running
### times will be as with default settings.
### Recover under the OUOU model.
OUOUestim<-ouchModel(phyltree,OUOUdata,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive")
```

```
### Recover under the OUBM model.
OUBMestim<-mvslouchModel(phyltree,OUBMdata,2,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive")

### Usage of the wrapper function that allows for estimation under multiple
### models in one call, here we use the default collection of models offered
### by mvSLOUCH. This includes BM, a number of OUOU and OUBM models.
### For data simulated under BM.
estimResultsBM<-estimate.evolutionary.model(phyltree,BMdata,regimes=NULL,
root.regime=NULL,M.error=NULL,repeats=3,model.setups=NULL,predictors=c(3),
kY=2,doPrint=TRUE)
### For data simulated under OUOU.
estimResultsOUOU<-estimate.evolutionary.model(phyltree,OUOUdata,regimes=regimes,
root.regime="small",M.error=NULL,repeats=3,model.setups=NULL,predictors=c(3),
kY=2,doPrint=TRUE)
### For data simulated under OUBM.
estimResultsOUBM<-estimate.evolutionary.model(phyltree,OUBMdata,regimes=regimes,
root.regime="small",M.error=NULL,repeats=3,model.setups=NULL,predictors=c(3),
kY=2,doPrint=TRUE)
## In the wrapper function the resulting best found model parameters are in
## estimResultsBM$BestModel$ParamsInModel
## estimResultsOUOU$BestModel$ParamsInModel
## estimResultsOUBM$BestModel$ParamsInModel

### Summarize the best found models.
### Under the BM model for BM data. One needs to check whether the
### model in BMestim$ParamsInModel corresponds to a BM model (here it does).
BM.summary<-SummarizeBM(phyltree,BMdata,BMestim$ParamsInModel,t=c(1),
dof=BMestim$ParamSummary$dof)
### Under the OUOU model for OUOU data. One needs to check whether the
### model in OUOUestim$ParamsInModel corresponds to an OUOU model (here it does).
OUOU.summary<-SummarizeOUCH(phyltree,OUOUdata,OUOUestim$FinalFound$ParamsInModel,
regimes,t=c(1),dof=OUOUestim$FinalFound$ParamSummary$dof)
### Under the OUBM model for OUBM data. One needs to check whether the
### model in OUBMestim$ParamsInModel corresponds to a OUBM model (here it does).
OUBM.summary<-SummarizeMVSLOUCH(phyltree,OUBMdata,OUBMestim$FinalFound$ParamsInModel,
regimes,t=c(1),dof=OUBMestim$FinalFound$ParamSummary$dof)


### Now run the parametric bootstrap to obtain confidence intervals for some parameters.
### For the BM model, we want CIs for the ancestral state, and the "sqaure" of the
### diffusion (evolutionary rate) matrix. The number of bootstrap replicates
### is very small, 5 (to reduce running times). In reality it should be much more, e.g.,
### 100 or more if the computational budget allows.
BMbootstrap<-parametric.bootstrap(estimated.model=BMestim,phyltree=phyltree,
values.to.bootstrap=c("vX0","StS"),M.error=NULL,numboot=5)
### For the OUOU model, we want a CI for the evolutionary regression coefficient vector.
### The number of bootstrap replicates is very small, 5 (to reduce running times).
### In reality it should be much more, e.g., 100 or more if the
### computational budget allows.
OUOUbootstrap<-parametric.bootstrap(estimated.model=estimResultsOUOU,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression"),regimes=regimes,root.regime="small",
M.error=NULL,predictors=c(3),kY=NULL,numboot=5,Atype=NULL,Syytype=NULL,diagA=NULL)
```

```
### For the OUBM model, we want CIs for the evolutionary and optimal regression
### coefficient vectors. The number of bootstrap replicates is very small,
### 5 (to reduce running times). In reality it should be much more, e.g.,
### 100 or more if the computational budget allows.
OUBMbootstrap<-parametric.bootstrap(estimated.model=OUBMestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression","optimal.regression"),
regimes=regimes,root.regime="small",M.error=NULL,predictors=c(3),kY=2,
numboot=5,Atype="DecomposablePositive",Syytype="UpperTri",diagA="Positive")

## End(Not run)
```

`BrownianMotionModel` *Estimate parameters under a Brownian motion model of evolution*

Description

The `BrownianMotionModel` function uses maximum likelihood to fit parameters of a Brownian motion model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
BrownianMotionModel(phyltree, mData, predictors = NULL, M.error = NULL,
min_bl = 0.0003)
```

Arguments

`phyltree` The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

`mData` A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

`predictors` A vector giving the numbers of the columns from `data` which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided, then none will be considered to be predictors.

`M.error` An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,

- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`min_bl` Value to which **PCMBase**'s `PCMBase.Threshold.Skip.Singular` should be set. It indicates that branches of length shorter than `min_bl` should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

This function estimates the parameters of a multivariate Brownian motion model defined by the SDE,

$$dX(t) = \Sigma dB(t), X(0) = X_0$$

evolving on a phylogenetic tree.

Without measurement error the parameters are obtained analytically via a GLS procedure. If measurement error is present, then the parameters are optimized over using `optim()`. The initial conditions for the optimization are motivated by Bartoszek & Sagitov (2015)'s univariate results.

From version `2.0.0` of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to `mData`.

The `phyltree_paths()` function enhances the tree for usage by `mvSLOUCH`. Hence, to save time, it is advisable to first do `phyltree<-mvSLOUCH::phyltree_paths(phyltree)` and only then use it with `BrownianMotionModel()`.

From version `2.0.0` of **mvSLOUCH** the parameter `calcCI` has been removed. The package now offers the possibility of bootstrap confidence intervals, see function `parametric.bootstrap`.

Value

ParamsInModel
: A list with estimated model parameters. The elements are vX0 : the ancestral trait, and Sxx where $t\Sigma_{xx}\Sigma_{xx}^T$ is the Brownian motion's covariance matrix at time t.

ParamSummary
: A list with summary statistics with elements, StS the infinitesimal covariance matrix $\Sigma_{xx}\Sigma_{xx}^T$, LogLik the log–likelihood, dof the degrees of freedom, m2loglik is -2log–likelihood, aic is the Akaike information criterion, aic.c is the Akaike information criterion corrected for small sample size, sic is the Schwarz information criterion, bic is the Bayesian information criterion (which is the same as the Schwarz information criterion) and RSS is the residual sum of squares.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Sagitov S. (2015) A consistent estimator of the evolutionary rate. Journal of Theoretical Biology 371:69-78.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Felsenstein, J. (1985) Phylogenies and the comparative method. American Naturalist 125:1-15.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Mitov, V. and Bartoszek, K. and Asimomitis, G. and Stadler, T. (2020) Fast likelihood calculation for multivariate Gaussian phylogenetic models with shifts Theoretical Population Biology 131:66-78.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

brown,mvBM, PCMLik, SummarizeBM, simulBMProcPhylTree, parametric.bootstrap

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree  using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
```

```
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define Brownian motion parameters to be able to simulate data under
### the Brownian motion model.
BMparameters<-list(vX0=matrix(0,nrow=3,ncol=1),
Sxx=rbind(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))

### Now simulate the data.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx)
BMdata<-BMdata[phyltree$tip.label,,drop=FALSE]

### Recover the parameters of the Brownian motion.
BMestim<-BrownianMotionModel(phyltree,BMdata)

## Not run:
### And finally obtain bootstrap confidence intervals for some parameters
BMbootstrap<-parametric.bootstrap(estimated.model=BMestim,phyltree=phyltree,
values.to.bootstrap=c("vX0","StS"),M.error=NULL,numboot=2)

## End(Not run)
RNGversion(as.character(getRversion()))
```

`drawPhylProcess` *Plots the realization of a process evolving on a phylogenetic tree*

Description

The function takes the output of the simulation functions and based on it plots the realization of the process on the tree. Can handle multiple traits, in this case each trait is plotted separately. The function does draw anything else (like axes) but the realization of the process. Any additions are up to the user.

Usage

```
drawPhylProcess(PhylTraitProcess, vTraitsToPlot=NULL, vColours = "black",
plotlayout = c(1, 1), additionalfigs = FALSE, modelParams = NULL,
EvolModel = NULL, xlimits = NULL, ylimits = NULL)
```

Arguments

`PhylTraitProcess`
: The simulated realization of the process, the direct output of one of the package's simulation function or a matrix (if `fullTrajectory` is `TRUE`). In the second case the matrix consists of k+1 columns, where k is the number of traits. The first column are the time instances, the next k the values of the traits at that instance.

Since evolution takes place on a phylogenetic tree - there should be multiple copies of the same time moment, i.e. one for each branch of the tree.

`vTraitsToPlot` A vector providing the column numbers of the traits to plot. If `NULL`, then all traits are plotted. The column numbers have to be obtained from the `PhylTraitProcess` object, the matrix `$trajectory` for each branch. Notice that the first column is time! The same trait may be plotted multiple times (but a warning will be raised).

`vColours` A vector of colours to be used for each trait. If length is less than the number of traits then colours are recycled

`plotlayout` How many plots per page if more than one trait, i.e. `par(mfrow=plotlayout))`.

`additionalfigs` Should additional items be plotted on each figure, the ancestral state and deterministic, Ψ when appropriate. If there are many regime levels then only the first column of Ψ is used.

`modelParams` List of model parameters.

`EvolModel` The evolutionary model.

`xlimits` The x limits of the plot. Can be useful to fix if one wants to have a number of graphs on the same scale. This can be either a vector of length 2 (minimum and maximum value of the x-axis), or a list of length equalling the number of traits with each entry being a vector of length 2 or a matrix with two columns and rows equalling the number of traits. If not provided then the value is just the minimum and maximum from the data.

`ylimits` The y limits of the plot. Can be useful to fix if one wants to have a number of graphs on the same scale. This can be either a vector of length 2 (minimum and maximum value of the x-axis), or a list of length equalling the number of traits with each entry being a vector of length 2 or a matrix with two columns and rows equalling the number of traits. If not provided then the value is just the minimum and maximum from the data.

Value

Returns a meaningless NA value.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
```

```
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(3)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
#regimes<-c("small","small","large","small","small","large","large","large")
#regimes<-c("small","small","large","small","small","large")
regimes<-c("small","small","large","small")

### Define SDE parameters to be able to simulate data under the OUOU model.
## 3D model
## OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
## A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
## "large"=c(-1,1,0.5)),Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

## 2D model for speed on CRAN
OUOUparameters<-list(vY0=matrix(c(1,-1),nrow=2,ncol=1),
A=rbind(c(9,0),c(0,5)),mPsi=cbind("small"=c(1,-1),
"large"=c(-1,1)),Syy=rbind(c(1,0.25),c(0,1)))


### Now simulate the data keeping the whole trajectory
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL,fullTrajectory=TRUE)
drawPhylProcess(PhylTraitProcess=OUOUdata,plotlayout=c(1,3))
RNGversion(as.character(getRversion()))
```

`estimate.evolutionary.model`

Wrapper function to find best (out of BM, OU, OUOU, OUBM) fitting evolutionary model and estimate its parameters.

Description

The `estimate.evolutionary.model` function calls the `BrownianMotionModel`, `ouchModel` and `mvslouchModel` functions with different classes of evolutionary model parameters. It then compares the resulting estimates by the AICc (or BIC if AICc fails) and returns the best overall model. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
estimate.evolutionary.model(phyltree, mData, regimes = NULL,
root.regime = NULL, M.error = NULL, repeats = 5, model.setups = NULL,
predictors = NULL, kY = NULL, doPrint = FALSE, pESS=NULL,
estimate.root.state=FALSE, min_bl = 0.0003, maxiter=c(10,50,100))
```

Arguments

phyltree
: The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`.

mData
: A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

regimes
: A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`.If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

root.regime
: The regime at the root of the tree. If not given, then it is taken as the regime that is present on the root's daughter lineages and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

M.error
: An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

repeats
: How many starting points for the numerical maximum likelihood procedure should be tried for each model setup. On the first repeat for OUOU and OUBM modes the functions takes as the starting point (for `A` and `Syy`) values based on the sample covariance matrix estimate, motivated by Bartoszek & Sagitov (2015)'s univariate results.

model.setups
: What models to try when searching for the best evolutionary model. This field may remain `NULL`, in this situation the function generates using `.generate.basic.model.setups()` a basic list of models. Allowed values are

"basic" A basic list of models to try out is generated, defined using .generate.basic.model.setups(). This list should be usually enough.

"fundamental" A slightly extended list of models to try out is generated, defined using .generate.fund.model.setups(). Compared to "basic" a few more models are added.

"extended" An extension of the "fundamental" list of models to try out. Defined using .generate.ext.model.setups() which at the moment calls generate.model.setups().

"all" All possible models are generated, using .generate.all.model.setups(). Running it will take an intolerable amount of time.

Alternatively the user is also free to provide their own list of models in this variable. Each element of the list is a list with fields.

evolmodel The evolutionary model, it may take one of the three values "bm" (Brownian motion model), "ouch" (OUOU model), "mvslouch" (OUBM model).

Atype The class of the A matrix, ignored if evolmodel equals "bm". Otherwise it can take one of the following values: "SingleValueDiagonal", "Diagonal", "UpperTri", "LowerTri", "SymmetricPositiveDefinite", "Symmetric", "DecomposablePositive", "DecomposableNegative", "DecomposableReal", "Invertible", "Any".

Syytype The class of the Syy matrix, ignored if evolmodel equals "bm". Otherwise it can take one of the following values: "SingleValueDiagonal", "Diagonal", "UpperTri", "LowerTri", "Symmetric", "Any".

diagA Should the diagonal of A be forced to be positive ("Positive"), negative ("Negative") or the sign free to vary (NULL). However, setting this to a non-NULL value when evolmodel is "mvslouch" might be (but simulations concerning this are not conclusive) slightly detrimental to the optimization process if Atype is "DecomposablePositive", "DecomposableNegative", or "DecomposableReal". In these cases A is parametrized by its eigendecomposition. Additional exponentiation of the diagonal, to ensure positivity, could (but this is uncertain) make the exploration of the likelihood surface more difficult. The user is advised to also try diag=NULL. In the case of Atype being "SymmetricPositiveDefinite", the diagonal is always guaranteed to be positive.

signsA WARNING: ONLY use this if you know what you are doing. Ignored if evolmodel equals "bm". This allows the user to specify which elements of A are to be positive, negative or equal to specific values.See ouchModel and mvslouchModel for a more specific description and important warnings.

signsSyy WARNING: ONLY use this if you know what you are doing. Ignored if evolmodel equals "bm". This allows the user to specify which elements of Syy are to be positive, negative or equal to specific values. See ouchModel and mvslouchModel for a more specific description and important warnings.

signsB WARNING: ONLY use this if you know what you are doing. Ignored if evolmodel does not equals "mvslouch". This allows the user to specify which elements of B are to be positive, negative or equal to specific values. See mvslouchModel for a more specific description and important warnings.

signsmPsi WARNING: ONLY use this if you know what you are doing. Ignored if evolmodel equals "bm". This allows the user to specify which elements of mPsi are to be positive, negative or equal to specific values.See ouchModel and mvslouchModel for a more specific description and important warnings.

signsvY0 WARNING: ONLY use this if you know what you are doing. Ignored if evolmodel equals "bm". This allows the user to specify which elements of vY0 are to be positive, negative or equal to specific values.See ouchModel and mvslouchModel for a more specific description and important warnings.

start_point_for_optim A named list with starting parameters for of the parameters for be optimized by optim(), currently only A and Syy for evolmodel equalling "ouch" or "mvslouch". One may provide both or only one of them. Make sure that the parameter is consistent with the other parameter restrictions as no check is done and this can result in undefined behaviour. For example one may provide this as (provided dimensions and other parameter restrictions agree)

start_point_for_optim=list(A=rbind(c(2,0),(0,4)), Syy=rbind(c(1,0.5),c(0,2))).

parscale A vector to calculate the parscale argument for optim. It is a named vector with 3 entries, e.g. c("parscale_A"=3,"logparscale_A"=5,"logparscale_other"=1). The entry parscale_A is the scale for entries of the A matrix, logparscale_A is the scale for entries of the A matrix that are optimized over on the logarithmic scale, e.g. if eigenvalues are assumed to be positive, then optimization is done over log(eigenvalue) for A's eigendecomposition and logparscale_other is the scale for entries other then of A that are done on the logarithmic scale (e.g. Syy's diagonal, or other entries indicated as positive via parameter_signs). If not provided (or if a name of the vector is misspelled), then made equal to the example value provided above. For other elements, then mentioned above, that are optimized over by optim(), 1 is used for optim()'s parscale. It is advised that the user experiments with a couple of different values and reads optim's man page.

estimateBmethod Only relevant for the OUBM models (optional), should B be estimated by maximum likelihood (default if not provided) value "ML" or generalized least squares (value "GLS").

A minimum example list is list(list(evolmodel="bm")). The functions that automatically generate different types of models do NOT use any of the "signs" parameters. Hence, in these models all parameters (under the appropriate parametrization) will be free to vary.

predictors A vector giving the numbers of the columns from dfdata which are to be considered predictor ones, *i.e.* conditioned on in the program output. A vector giving the numbers of the columns from mData matrix which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided then in for the OUBM model the columns `(kY+1):ncol(mData)`, i.e. the "BM" ones, are treated as predictors. Otherwise, none will be considered to be predictors.

`kY` Number of "Y" (response) variables, for the OUBM models. The first `kY` columns of `mY` are the "OU" ones, while the rest the "BM" ones. In more detail this value determines the number of columns of the `mData` matrix to treat as response variables ("OU" ones). For example, a value of 1 means that only the first column is treated as a response variable, while a value of 3 means the first three columns are treated as response variables. Any predictor variables ("BM" ones) the user is interested in setting for a particular model should therefore be placed in the final columns of the `mData` matrix, allowing for selecting select `kY` columns before this as response variables ("OU" ones).

`doPrint` Should the function print out information on what it is doing (TRUE) or keep silent (default FALSE).

`pESS` Should the function also find the best model taking into account the phylogenetic effective sample size and it so what method. If `NULL`, then do not take this into account. Otherwise one of `"reg"` ("regression" effective sample size that takes into account all of the correlations between species explicitly), `"mean"` (mean value effective sample size $1^T R^{-1} 1$, where R is the interspecies correlation matrix), `"MI"` (mutual information effective sample size), `"mvreg"`(multivariate version of "regression" effective sample size when each species is described by a suite of traits) , `"mvMI"` (multivariate mutual information effective sample size when each species is described by a suite of traits) indicating the way to calculate the pESS. The default (`NULL`) is not to do any pESS calculations as these will be slow. They require the construction of the between-species-between-traits variance covariance matrix and hence do not fully take advantage of the speed-up offered by **PCMBase**. If `pESS="only_calculate"`, then all possible pESS values are calculated but no model selection is done based on them.

`estimate.root.state`
Should the root state be estimate `TRUE` (not recommended) or set at the optimum `FALSE` (recommended). Root state estimation is usually unreliable hence if fossil measurements are available prediction based on them and the estimated model will probably be more accurate. If there is only one regime, then estimation of the root state separately is impossible and will not be allowed.

`min_bl` Value to which **PCMBase**'s `PCMBase.Threshold.Skip.Singular` should be set. It indicates that branches of length shorter than `min_bl` should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. Furthermore, **mvSLOUCH** sets all branches in the tree shorter than `min_bl` to `min_bl`. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

`maxiter` The maximum number of iterations for different components of the estimation algorithm. A vector of three integers. The first is the number of iterations for phylogenetic GLS evaluations, i.e. conditional on the other parameters, the regime optima, perhaps `B`, and perhaps initial state are estimated by a phylogenetic GLS procedure. After this the other (except of `B` in OUBM model case) parameters are optimized over by `optim()`. This first entry controls the number of iterations of this procedure. The second is the number of iterations inside the iterated GLS for the OUBM model. In the first step regime optima and `B` (and perhaps initial state) are estimated conditional on the other parameters and current estimate of `B`, then the estimate of `B` is update and the same phylogenetic GLS is repeated (second entry of `maxiter` number of times). Finally, the third is the value of `maxiter` passed to `optim()`, apart from the optimization in the Brownian motion and measurement error case.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from `https://github.com/venelin/PCMBaseCpp`.

The setting `Atype="Any"` means that one assumes the matrix `A` is eigendecomposable. If the estimation algorithm hits a defective `A`, then it sets the log-likelihood at the minimum value and will try to get out of this dip.

If `model.setups` is left at the default value the function will take a long time to run, as it performs estimation for each model (`generate.model.setups` generates 90 setups) times the value in repeats. Therefore if the user has particular hypotheses in mind then it is advisable to prepare their own list. If the `Syy` matrix is assumed to be upper-triangular and the starting conditions based on Bartoszek & Sagitov (2015)'s results are used then the factorization of $\Sigma = \Sigma_{yy}\Sigma_{yy}^T$ into Σ_{yy} is done using the procedure described in `https://math.stackexchange.com/questions/2039477/cholesky-decompostion-upper-triangular-or-lower-triangular`.

From version `2.0.0` of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to `mData`.

If AICc fails, then the function will use BIC to select between models. This is extremely unlikely essentially only when AICc is infinite, i.e. the model is saturated (number of observations equals number of data points).

Value

A list is returned that describes the results of the search. See the help for `BrownianMotionModel`, `ouchModel` and `mvslouchModel` for the description of the lower level entries. The elements of this list are the following

`BestModel` The resulting best model found. Included are the model parameters, a "first-glance" qualitative description of the model, the most important parameters of the process (half-lives and regressions in the case of OU models) and what to call to obtain standard errors. It takes a long time to obtain them so calculating them is not part of the standard procedure.

`BestModelESS` Only if `pESS` was `TRUE`. The resulting best model found taking into account the phylogenetic essential sample size. Included are the model parameters, a "first-glance" qualitative description of the model, the most important parameters of the process (half-lives and regressions in the case of OU models) and what to call to obtain bootstrap confidence intervals. It takes a long time to obtain them so calculating them is not part of the standard procedure.

`testedModels` A list of results for each tried model.

`model.setups` A list of models tried.

`repeats` How many starting points were tried per model.

Note

The engine behind the likelihood calculations is called from **PCMBase**. The `slouch` package is a recommended alternative if one has a OUBM models and only a single response (Y) trait. The `mvMORPH`, `ouch` and `Rphylpars` packages consider multivariate OU models and looking at them could be helpful.

Author(s)

Krzysztof Bartoszek

References

Ane, C. (2008) Analysis of comparative data with hierarchical autocorrelation. Annals of Applied Statistics 2:1078-1102.

Bartoszek, K. (2016) Phylogenetic effective sample size. Journal of Theoretical Biology 407:371-386.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Sagitov, S. (2015) Phylogenetic confidence intervals for the optimal trait value. Journal of Applied Probability 52(4):1115-1132.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. and Pienaar, J. and Orzack, S.H. (2008) A comparative method for studying adaptation to randomly evolving environment. Evolution 62:1965-1977.

Mitov, V. and Bartoszek, K. and Asimomitis, G. and Stadler, T. (2020) Fast likelihood calculation for multivariate Gaussian phylogenetic models with shifts Theoretical Population Biology 131:66-78.

Xiao, H and Bartoszek, K. and Lio P. (2018) Multi–omic analysis of signalling factors in inflammatory comorbidities. BMC Bioinformatics, Proceedings from the 12th International BBCC conference 19:439.

See Also

brown, mvBMBrownianMotionModel, SummarizeBM, simulBMProcPhylTree, hansen, mvOU, ouchModel, SummarizeOUCH, simulOUCHProcPhylTree, slouch::model.fit, PCMLik, mvslouchModel, SummarizeMVSLOUCH, simulMVSLOUCHProcPhylTree, parametric.bootstrap, optim

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(4)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","large","small")

### Define SDE parameters to be able to simulate data under the OUOU model.
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

## set up for a trivial, single model setup case (for running time)
## in a real analysis you should carefully choose between what models
## you want to do model selection
model_setups<-list(list(evolmodel="bm"))

### Try to recover the parameters of the OUOU model.
### maxiter here set to minimal working possibility, in reality it should be larger
### e.g. default of c(10,50,100)
estimResults<-estimate.evolutionary.model(phyltree,OUOUdata,regimes=regimes,
root.regime="small",M.error=NULL,repeats=1,model.setups=model_setups,predictors=c(3),
kY=2,doPrint=TRUE,pESS=NULL,maxiter=c(1,1,1))

### After this step you can look at the best estimated model and use the
### parametric.bootstrap() function to obtain bootstrap confidence intervals
RNGversion(as.character(getRversion()))

## Not run:  ##It takes too long to run this
## take a less trivial setup
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)
```

```
### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the OUOU model.
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

## set up for two very simple (for example usage) models to compare between
## in a real analysis you should carefully choose between what models
## you want to do model selection, the default
## model_setups<-NULL provides a wide selection of models

model_setups<-list(list(evolmodel="bm"),list(evolmodel="ouch",
"Atype"="SingleValueDiagonal","Syytype"="SingleValueDiagonal","diagA"="Positive"))

### Try to recover the parameters of the OUOU model.
estimResults<-estimate.evolutionary.model(phyltree,OUOUdata,regimes=regimes,
root.regime="small",M.error=NULL,repeats=3,model.setups=model_setups,predictors=c(3),
kY=2,doPrint=TRUE,pESS=NULL,maxiter=c(10,50,100))

## End(Not run)
```

`fitch.mvsl` *Unordered Fitch parsimony reconstruction of discrete character states*

Description

Implements an unordered Fitch parsimony reconstruction of discrete niche variables for use in the OU models where optima are modeled on discrete, categorical niche encodings. Allows for delayed and accelerated transformations to deal with ambiguities. Function was originally the `fitch()` function from the **slouch** package.

Usage

```
fitch.mvsl(phyltree, niche, deltran = FALSE, acctran = FALSE, root = NULL)
```

Arguments

`phyltree`	The phylogenetic tree in **ape** (`phylo`). For a phylogeny in `phylo` format the "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`.
`niche`	The specific niche variable in the **slouch** data.frame to be reconstructed, entered as data.frame$niche . The order of the niche's regimes has to correspond to the order of the tip nodes in `phyltree`.

deltran	Implements a delayed transformation algorithm in order to deal with ambiguous nodes
acctran	Implements an accelerated transformation algorithm to deal with ambiguous nodes
root	An optional argument allowing the user to define a character state for the root (useful if the root node is ambiguously reconstructed)

Details

The fitch.mvsl function is meant to be interactive, where the user acts on the advice given in the returned messages whilst attempting to reconstruct ancestral states. If the root node is ambiguous after an initial reconstruction (a message will be printed to the screen if this is the case), this needs to be set by the user using the root = "state" argument in the function call. Any remaining ambiguous nodes can then be dealt with by specifying deltran or acctran ="TRUE" in the function call

Value

The fitch.mvsl function returns a list with two or three elements. The first, $branch_regimes is a vector of reconstructed character states. Each entry of the vector corresponds to the respective edge in the $edge field in the provided tree. Notice that entries correspond to edges and not to nodes. If you require correspondence with nodes, then you can treat the given edge entry as the value for the node ending the edge. Actually, this is what the algorithm in the function estimates. The second field of the output object, $root_regimes is the regime at the root of the tree. If the provided tree was a raw phylo object, then the function will also return an enhanced version of it (field $phyltree). This is the tree that results from calling **mvSLOUCH**::phyltree_paths(phyltree) on the originally provided tree. This enhanced version is returns as calculating it is costly and the user might want to re-use it in some downstream analysis with **mvSLOUCH**. All **mvSLOUCH** user-level functions first enhance the provided phylogeny by **mvSLOUCH**::phyltree_paths(), but they first check if it is not already enhanced.

Author(s)

Jason Pienaar <jasonpienaar@gmail.com>

References

Fitch, M.W. (1971) Defining the course of Evolution: Minimum change for a specific tree topology. *Systematic Zoology* **20**:406–416.

Swofford, D. L. and W.P. Maddison (1987) Reconstructing ancestral character states under Wagner parsimony. *Mathematical Biosciences* **87**: 199–229.

See Also

slouch::fitch, slouch::slouchtree.plot, slouch::model.fit, slouch::ouch2slouch

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
phyltree<-ape::rtree(5)

regimes<-c("A","B","B","C","C")
regimesFitch<-fitch.mvsl(phyltree,regimes,root=1,deltran=TRUE)
RNGversion(as.character(getRversion()))
```

---

generate.model.setups *Generate a list of model setups for the function* estimate.evolutionary.model.

Description

The function generates a list of models that will be used by the function estimate.evolutionary.model. A minimum example list will be `list(list(evolmodel="bm"))`.

Usage

```
generate.model.setups()
```

Details

The function should really be a hidden one but is left available for the user as an example how such a list of models should be generated.

The setting `Atype="Any"` means that one assumes the matrix `A` is eigendecomposable. If `A` is defective, then the output will be erroneous.

None of the "signs" options for the model is generated, see the description of `mvslouchModel` and `ouchModel`.

Value

A list with different models is returned. Each element of the list is a list with the following fields.

`evolmodel` The evolutionary model, it may take one of the three values `"BM"` (Brownian motion model), `"ouch"` (OUOU model), `"mvslouch"` (OUBM model).

`Atype` The class of the `A` matrix, ignored if `evolmodel` equals `"BM"`. Otherwise it can take one of the following values: `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"SymmetricPositiveDefinite"`, `"Symmetric"`, `"DecomposablePositive"`, `"DecomposableNegative"`, `"DecomposableReal"`, `"Invertible"`, `"TwoByTwo"`, `"Any"`.

`Syytype` The class of the `A` matrix, ignored if `evolmodel` equals `"BM"`. Otherwise it can take one of the following values: `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"Any"`.

`diagA` Should the diagonal of `A` be forced to be positive (`TRUE`), negative (`FALSE`) or the sign free to vary (`NULL`)

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

See Also

`estimate.evolutionary.model`, `mvslouchModel`, `ouchModel`

Examples

```
model_setups<-generate.model.setups()
```

`mvslouchModel` *Estimate parameters under a (multivariate) OUBM model of evolution*

Description

The `mvslouchModel` function uses maximum likelihood to fit parameters of a multivariate OUBM model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
mvslouchModel(phyltree, mData, kY, regimes = NULL, regimes.times = NULL,
root.regime = NULL, predictors = NULL, M.error = NULL, Atype = "Invertible",
Syytype = "UpperTri", diagA = "Positive", estimate.root.state=FALSE,
parameter_signs=NULL, start_point_for_optim = NULL, parscale = NULL,
min_bl = 0.0003, maxiter = c(10,50,100), estimateBmethod="ML")
```

Arguments

`phyltree` The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

`mData` A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

`kY` Number of "Y" (response) variables. The first `kY` columns of `mY` are the "OU" ones, while the rest the "BM" ones. In more detail this value determines the number of columns of the `mData` matrix to treat as response variables ("OU" ones). For example, a value of 1 means that only the first column is treated as a response variable, while a value of 3 means the first three columns are treated as response variables. Any predictor variables ("BM" ones) the user is interested in setting for a particular model should therefore be placed in the final columns of the `mData` matrix, allowing for selecting select `kY` columns before this as response variables ("OU" ones).

`regimes` A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

`regimes.times` A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

`root.regime` The regime at the root of the tree. If not given, then it is taken as the regime that is present on the root's daughter lineages and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

`predictors` A vector giving the numbers of the columns from `mData` matrix which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided then columns `(kY+1):ncol(mData)`, i.e. the "BM" ones, are treated as predictors.

`M.error` An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,

- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`Atype` What class does the A matrix in the multivariate OUBM model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"SymmetricPositiveDefinite"`, `"DecomposablePositive"`, `"DecomposableNegative"`, `"DecomposableReal"`, `"Invertible"`, `"TwoByTwo"`, `"Any"`

`Syytype` What class does the Syy matrix in the multivariate OUBM model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"Any"`

`diagA` Whether the values on `A`'s diagonal are to be `"Positive"`, `"Negative"` or sign allowed to vary, `NULL`. However, setting this to a non-`NULL` value might be (but simulations concerning this are not conclusive) slightly detrimental to the optimization process if `Atype` is `"DecomposablePositive"`, `"DecomposableNegative"`, or `"DecomposableReal"`. In these cases `A` is parametrized by its eigendecomposition. Additional exponentiation of the diagonal, to ensure positivity, could (but this is uncertain) make the exploration of the likelihood surface more difficult. The user is advised to also try `diag=NULL`. In the case of `Atype` being `"SymmetricPositiveDefinite"`, the diagonal is always guaranteed to be positive.

`estimate.root.state`
Should the root state be estimate `TRUE` (not recommended) or set at the optimum `FALSE` (recommended). Root state estimation is usually unreliable hence if fossil measurements are available prediction based on them and the estimated model will probably be more accurate. If there is only one regime, then estimation of the root state separately is impossible and will not be allowed.

`parameter_signs`
WARNING: ONLY use this option if you understand what you are doing! This option is still in an experimental stage so some setups might not work (please report). A list allowing the user to control whether specific entries for each model parameter should be positive, negative, zero or set to a specific (other) value. The entries of the list have to be named, the admissible names are `"signsA"` (for `A` matrix), `"signsB"` (for `B` matrix), `"signsSyy"` (for `Syy` matrix) and `"signsmPsi"` (for `mPsi` matrix) and `"signsvY0"` (for `vY0` matrix). Any other entry in this list will be ignored. Each entry of the list has to be a matrix of appropriate size, i.e. of the size of the parameter to which it corresponds. Inside this matrix the possible values are `"+"` if the given entry is to be positive, `"-"` if the given entry is to be negative, `x`, where `x` is a number, if the entry is to be set to specified value or `NA` if the entry is to be freely estimated. See Details for an example, further description and important warnings!

`start_point_for_optim`
: A name list with starting parameters for of the parameters for be optimized by `optim()`, in this case `A` and `Syy`. One may provide both or only one of them. Make sure that the parameter is consistent with the other parameter restrictions as no check is done and this can result in undefined behaviour. For example one may provide this as (provided dimensions and other parameter restrictions agree)

```
start_point_for_optim=list(A=rbind(c(2,0),(0,4)),
Syy=rbind(c(1,0.5),c(0,2))).
```

`parscale`
: A vector to calculate the `parscale` argument for `optim`. It is a named vector with 3 entries, e.g. `c("parscale_A"=3,"logparscale_A"=5,"logparscale_other"=1)`. The entry `parscale_A` is the scale for entries of the `A` matrix, `logparscale_A` is the scale for entries of the `A` matrix that are optimized over on the logarithmic scale, e.g. if eigenvalues are assumed to be positive, then optimization is done over `log(eigenvalue)` for `A`'s eigendecomposition and `logparscale_other` is the scale for entries other then of `A` that are done on the logarithmic scale (e.g. `Syy`'s diagonal, or other entries indicated as positive via `parameter_signs`). If not provided (or if a name of the vector is misspelled), then made equal to the example value provided above. For other elements, then mentioned above, that are optimized over by `optim()`, `1` is used for `optim()`'s `parscale`. It is advised that the user experiments with a couple of different values and reads `optim`'s man page.

`min_bl`
: Value to which **PCMBase**'s `PCMBase.Threshold.Skip.Singular` should be set. It indicates that branches of length shorter than `min_bl` should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

`maxiter`
: The maximum number of iterations for different components of the estimation algorithm. A vector of three integers. The first is the number of iterations for phylogenetic GLS evaluations, i.e. conditional on the other parameters, the regime optima, `B` and perhaps initial state are estimated by a phylogenetic GLS procedure. After this the other (except of `B`) parameters are optimized over by `optim()`. This first entry controls the number of iterations of this procedure. The second is the number of iterations inside the iterated GLS. In the first step regime optima and `B` (and perhaps initial state) are estimated conditional on the other parameters and current estimate of `B`, then the estimate of `B` is update and the same phylogenetic GLS is repeated (second entry of `maxiter` number of times). Finally, the third is the value of `maxiter` passed to `optim()`, apart from the optimization in the Brownian motion and measurement error case.

`estimateBmethod`
: Should `B` be estimated by maximum likelihood (default value `"ML"`) or generalized least squares (value `"GLS"`).

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ back-end, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

This function estimates the parameters of the following multivariate SDE,

$$\begin{array}{rclrcl} dY(t) & = & -A(Y(t)-(\Psi(t)-A^{-1}BX(t)))dt+\Sigma_{yy}dB(t) & Y(0) & = & Y_0, \\ dX(t) & = & \Sigma_{xx}dB(t) & X(0) & = & X_0 \end{array}$$

on a phylogenetic tree. It uses a numerical optimization over `A` (parametrized by its eigenvalues and eigenvectors or its QR decomposition) and `S` (parametrized by its values) and conditional on `A` and `S` estimates the values of `Psi` corresponding to the different regimes by a GLS estimate. `Y(0)` is assumed to be equal to `- solve(A)BX(0)` plus the root value of `Psi`. This assumes that `A` is invertible. If not, then `Y(0)` will be set at the root value of `Psi`. This is unless `estimate.root.state=TRUE`, in such a case `Y(0)` will be estimated by least squares.

The setting `Atype="Any"` means that one assumes the matrix `A` is eigendecomposable. If the estimation algorithm hits a defective `A`, then it sets the log-likelihood at the minimum value and will try to get out of this dip.

The function parameter `parameter_signs` is special in the sense that it can give the user great control over the estimation procedure but can also make the output very inconsistent with what the user provides. If we have two response traits (OU ones) and two predictor traits (BM ones), then an EXAMPLE setting of this can be:
`parameter_signs=list(signsA=rbind(c("+","-"),c(0,"+")),`
`signsSyy=rbind(c(NA,0),c(0,NA)), signsB=rbind(c(NA,0),c(0,NA)))`. This means that `A` is upper triangular with positive values on the diagonal and a negative value on the off-diagonal, `Syy` is diagonal and `B` is also diagonal. It is advisable to set now `Atype="Any"` and `Syytype="Any"` (see further description).

If the given model parameter is to be estimated by a generalized least squares (currently `B`, `mPsi` and `vY0`), then the sign specifications are ignored. However, it is possible to set specific values. Furthermore, the package does not check (for `A` and `Syy`) if the specifications here agree with the `Atype`, `Syytype` and `diagA`. The settings in `signsA` and `signsSyy` will override the other settings. Hence, it is up to the user to make sure that the settings of `signsA` and `signsSyy` are consistent with `Atype`, `Syytype` and `diagA`. It is advisable to use `signsA` with `"+"` on the diagonal and have `diagA=NULL`. The diagonal of `Syy` is forced to be positive (unless `"-"` is used on the diagonal of `signsSyy` but this is strongly discouraged) so it is advisable to keep `NA` on the diagonal of `signsSyy` and not put there `"+"` there. Hence, in particular using the signs mechanism result in a wrong class of the matrix (e.g. `Atype="SymmetricPositiveDefinite"`, but after corrections for the provided entries in `signsA` one obtains a non-symmetric `A` with complex, negative-real-part eigenvalues). Lastly, using `signsA` and `signsSyy` can result in a wrong amount of `dof` and in turn incorrect `AICc` and `BIC` values. What the code does is subtracts the amount of fixed values in `signsA` and `signsSyy` from the amount of free parameters used to estimate `A` and `Syy`. For example if one sets
`Atype="SingleValueDiagonal"` (estimated by one free parameter) but specified two off-diagonal values, then the amount of dofs from `A` will be `-1`!! The ONLY fail-safe way to use this is to set `Atype="Any"` (if `signsA` used) and `Syytype="Any"` (if `signsSyy` used). If using `Syytype="Any"` and `signsSyy` the it is strongly advisable to set the entries either below or above `Syy`'s diagonal to

`0`. The reason is that $\Sigma_{yy}\Sigma_{yy}^T$ enters the likelihood and not the given value of Σ_{yy}. Hence, having values below (or respectively above) the diagonal results in an overparameterized model. The package has the option of mixing different matrix types with specifying values in it but this is only for advanced users who need to dig into the code to see what the `dof`'s should be and if it is possible to find a correspondence between the parametrization and settings. If entries of `mPsi`, `vY0` and `B` are pre-specified, then the `dof` are correctly adjusted for this. The estimation procedures currently ignore any pre-specified values for `vX0` and `Sxx`!

The found point is described by a list containing four fields. The first field `HeuristicSearchPointFinalFind` is the parametrization of the model parameters at the considered point with the value of the log–likelihood. The field `ParamsInModel` is the point estimate of the parameters of the SDE. The field `ParamSummary` are different composite (evaluated at the tree's height) and summary statistics, The field `phylhalflife` are the eigenvalues, eigenvectors and phylogenetic half lives associated with the A matrix of, `expmtA` is $exp(-A*(treeheight))$, optimal regression is the $A^{-1}B$ matrix (if A is invertible, otherwise this will not exist), `mPsi.rotated` is each of the regime effects multiplied by $1-\exp(-A*(treeheight))$, `cov.matrix` is the trait vector covariance matrix at the tree's height, `corr.matrix` is the trait vector correlation matrix at the tree's height, `conditional.cov.matrix` is the conditional covariance matrix of the OU type variables on the Brownian motion type at the tree's height, i.e. Cov[Y|X](tree height), `conditional.corr.matrix` is the conditional correlation matrix of the OU type variables on the Brownian motion type at the tree's height, i.e. Corr[Y|X](tree height), `stationary.cov.matrix` is the limit of the `conditional.cov.matrix`, `stationary.corr.matrix` is the limit of the `conditional.corr.matrix`, `optima.cov.matrix` is the covariance matrix of the optimal process at the tree's height equalling $(treeheight)*A^{-1}B\Sigma_{xx}\Sigma_{xx}^T B^T A^{-T}$, `optima.corr.matrix` is the correlation matrix of the optimal process at time the tree's height, `cov.with.optima` is the covariance matrix between the optimal process and the Y type variables process, `corr.with.optima` is the correlation matrix between the optimal process and the Y type variables process, `evolutionary.regression` is the regression coefficient of E[Y|X](tree height). Everything concerning the optimal process assumes A has positive real-part eigenvalues (in particular it is invertible). Otherwise these will not exist. `StS` is the infinitesimal covariance matrix, `LogLik` the log–likelihood, dof the degrees of freedom, `m2loglik` is -2log–likelihood, `aic` is the Akaike information criterion, `aic.c` is the Akaike information criterion corrected for small sample size, sic is the Schwarz information criterion, `bic` is the Bayesian information criterion (which is the same as the Schwarz information criterion) and `RSS` is the residual sum of squares. The field `RSS_non_phylogenetic` is a residual sum of squares calculated without correcting for the phylogeny–induced between species correlations, while the extension conditional_on_predictors indicates that we consider the RSS for the variables labelled as responses conditioned on the remaining variables. The `R2_phylaverage` field is R2, where the alternative model is the phylogenetically weighted sample average (see `OU_phylreg`). The last field `LogLik` is the log–likelihood at the point.

From version `2.0.0` of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to `mData`.

From version `2.0.0` of **mvSLOUCH** the parameter `calcCI` has been removed. The package now offers the possibility of bootstrap confidence intervals, see function `parametric.bootstrap`.

Value

`FinalFound` The point where the search procedure stopped. See Details for the description.

MaxLikFound The point with the highest likelihood found by the search procedure, if it is the same as the final point then this field equals "Same as final found".

Warning

The estimation can take a long time and should be repeated a couple of times so that it is run from different starting positions. The function can produce (a lot of) warnings and errors during the search procedure, this is nothing to worry about.

Note

The slouch package is a recommended alternative if one has only a single response (Y) trait.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Hansen, T.F. and Pienaar, J. and Orzack, S.H. (2008) A comparative method for studying adaptation to randomly evolving environment. Evolution 62:1965-1977.

Labra, A., Pienaar, J. & Hansen, T.F. (2009) Evolution of thermophysiology in Liolaemus lizards: adaptation, phylogenetic inertia and niche tracking. The American Naturalist 174:204-220.

Mitov, V. and Bartoszek, K. and Asimomitis, G. and Stadler, T. (2020) Fast likelihood calculation for multivariate Gaussian phylogenetic models with shifts Theoretical Population Biology 131:66-78.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

PCMLik, slouch::model.fit, SummarizeMVSLOUCH, simulMVSLOUCHProcPhylTree, parametric.bootstrap, optim

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(3)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

## 2 regimes
### Define a vector of regimes.
## regimes<-c("small","small","large","small")
## OUBMparameters<-list(vY0=matrix(1,ncol=1,nrow=1),A=matrix(0.5,ncol=1,nrow=1),
## B=matrix(2,ncol=1,nrow=1),mPsi=cbind("small"=1,"large"=-1),
## Syy=matrix(2,ncol=1,nrow=1),vX0=matrix(0,ncol=1,nrow=1),Sxx=diag(2,1,1),
## Syx=matrix(0,ncol=1,nrow=1),Sxy=matrix(0,ncol=1,nrow=1))
## single regime for speed on CRAN
regimes<-c("small","small","small","small")
OUBMparameters<-list(vY0=matrix(1,ncol=1,nrow=1),A=matrix(0.5,ncol=1,nrow=1),
B=matrix(2,ncol=1,nrow=1),mPsi=cbind("small"=1),
Syy=matrix(2,ncol=1,nrow=1),vX0=matrix(0,ncol=1,nrow=1),Sxx=diag(2,1,1),
Syx=matrix(0,ncol=1,nrow=1),Sxy=matrix(0,ncol=1,nrow=1))


### Now simulate the data.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[phyltree$tip.label,,drop=FALSE]

### Try to recover the parameters of the mvOUBM model.
### maxiter here set to minimal working possibility, in reality it should be larger
### e.g. default of c(10,50,100)
### Also the Atype and Syytype variables should be changed, here set as simplest
### for speed of evaluation, e.g. Atype="DecomposablePositive", Syytype="UpperTri"
OUBMestim<-mvslouchModel(phyltree,OUBMdata,1,regimes,Atype="SingleValueDiagonal",
Syytype="SingleValueDiagonal",diagA="Positive",maxiter=c(1,2,1))
RNGversion(as.character(getRversion()))

## Not run:  ##It takes too long to run this
## take a less trivial setup
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the mvOUBM model.

OUBMparameters<-list(vY0=matrix(c(1,-1),ncol=1,nrow=2),A=rbind(c(9,0),c(0,5)),
```

```
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))


### Now simulate the data.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[phyltree$tip.label,,drop=FALSE]

### Try to recover the parameters of the mvOUBM model.
OUBMestim<-mvslouchModel(phyltree,OUBMdata,2,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive",maxiter=c(10,50,100))

### And finally bootstrap with particular interest in the evolutionary and optimal
### regressions
OUBMbootstrap<-parametric.bootstrap(estimated.model=OUBMestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression","optimal.regression"),
regimes=regimes,root.regime="small",M.error=NULL,predictors=c(3),kY=2,
numboot=5,Atype="DecomposablePositive",Syytype="UpperTri",diagA="Positive")

## End(Not run)
```

---

ouchModel *Estimate parameters under a (multivariate) OU model of evolution*

Description

The ouchModel function uses maximum likelihood to fit parameters of a multivariate OU model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
ouchModel(phyltree, mData, regimes = NULL, regimes.times = NULL,
root.regime = NULL, predictors = NULL, M.error = NULL, Atype = "Invertible",
Syytype = "UpperTri", diagA = "Positive", estimate.root.state = FALSE,
parameter_signs = NULL, start_point_for_optim = NULL, parscale = NULL,
min_bl = 0.0003, maxiter = c(10,100))
```

Arguments

phyltree The phylogeny in phylo format. The tree can be obtained from e.g. a nexus file by the read.nexus() function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices 1:n and the root index n+1. The root.edge field is ignored.

mData A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field phyltree$tip.label), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

`regimes` A vector or list of regimes. If vector then each entry corresponds to each of the branches of `phyltree`, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

`regimes.times` A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

`root.regime` The regime at the root of the tree. If not given, then it is taken as the regime that is present on the root's daughter lineages and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

`predictors` A vector giving the numbers of the columns from `data` which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided, then none will be treated as predictors.

`M.error` An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`Atype` What class does the A matrix in the multivariate OUOU model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"SymmetricPositiveDefinite"`, `"DecomposablePositive"`, `"DecomposableNegative"`, `"DecomposableReal"`, `"Invertible"`, `"TwoByTwo"`, `"Any"`

`Syytype` What class does the Syy matrix in the multivariate OUBM model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"Any"`

`diagA` Whether the values on `A`'s diagonal are to be `"Positive"`, `"Negative"` or sign allowed to vary, `NULL`. However, setting this to a non-`NULL` value might be detrimental to the optimization process if Atype is "SymmetricPositiveDefinite". In this case the diagonal is always guaranteed to be positive.

estimate.root.state
: Should the root state be estimate TRUE (not recommended) or set at the optimum FALSE (recommended). Root state estimation is usually unreliable hence if fossil measurements are available prediction based on them and the estimated model will probably be more accurate. If there is only one regime, then estimation of the root state separately is impossible and will not be allowed.

parameter_signs
: WARNING: ONLY use this option if you understand what you are doing! This option is still in an experimental stage so some setups might not work (please report). A list allowing the user to control whether specific entries for each model parameter should be positive, negative, zero or set to a specific (other) value. The entries of the list have to be named, the admissible names are "signsA" (for A matrix), "signsSyy" (for Syy matrix) and "signsmPsi" (for mPsi matrix) and "signsvY0" (for vY0 matrix). Any other entry in this list will be ignored. Each entry of the list has to be a matrix of appropriate size, i.e. of the size of the parameter to which it corresponds. Inside this matrix the possible values are "+" if the given entry is to be positive, "-" if the given entry is to be negative, x, where x is a number, if the entry is to be set to specified value or NA if the entry is to be freely estimated. See Details for an example, further description and important warnings!

start_point_for_optim
: A named list with starting parameters for of the parameters for be optimized by optim(), in this case A and Syy. One may provide both or only one of them. Make sure that the parameter is consistent with the other parameter restrictions as no check is done and this can result in undefined behaviour. For example one may provide this as (provided dimensions and other parameter restrictions agree)

```
start_point_for_optim=list(A=rbind(c(2,0),(0,4)),
Syy=rbind(c(1,0.5),c(0,2))).
```

parscale
: A vector to calculate the parscale argument for optim. It is a named vector with 3 entries, e.g.
c("parscale_A"=3,"logparscale_A"=5,"logparscale_other"=1). The entry parscale_A is the scale for entries of the A matrix, logparscale_A is the scale for entries of the A matrix that are optimized over on the logarithmic scale, e.g. if eigenvalues are assumed to be positive, then optimization is done over log(eigenvalue) for A's eigendecomposition and logparscale_other is the scale for entries other then of A that are done on the logarithmic scale (e.g. Syy's diagonal, or other entries indicated as positive via parameter_signs). If not provided (or if a name of the vector is misspelled), then made equal to the example value provided above. For other elements, then mentioned above, that are optimized over by optim(), 1 is used for optim()'s parscale. It is advised that the user experiments with a couple of different values and reads optim's man page.

min_bl
: Value to which **PCMBase**'s PCMBase.Threshold.Skip.Singular should be set. It indicates that branches of length shorter than min_bl should be skipped

in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

`maxiter` The maximum number of iterations for different components of the estimation algorithm. A vector of two integers. The first is the number of iterations for phylogenetic GLS evaluations, i.e. conditional on the other parameters, the regime optima and perhaps initial state are estimated by a phylogenetic GLS procedure. After this the other parameters are optimized over by `optim()`. This first entry controls the number of iterations of this procedure. Finally, the second is the value of `maxiter` passed to `optim()`.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

This function estimates the parameters of the following multivariate SDE,

$$dY(t) = -A(Y - \Psi(t))dt + \Sigma dW(t), Y(0) = Y_0$$

on a phylogenetic tree. It uses a numerical optimization over `A` (parametrized by its eigenvalues and eigenvectors or its QR decomposition) and `S` (parametrized by its values) and conditional on `A` and `S` estimates the values of Psi corresponding to the different regimes by a GLS estimate. `Y(0)` is assumed to be equal to the root value of `Psi` (unless `estimate.root.state=TRUE`), then `Y(0)` is estimated is estimated by least squares).

The setting `Atype="Any"` means that one assumes the matrix `A` is eigendecomposable. If the estimation algorithm hits a defective `A`, then it sets the log-likelihood at the minimum value and will try to get out of this dip.

The function parameter `parameter_signs` is special in the sense that it can give the user great control over the estimation procedure but can also make the output very inconsistent with what the user provides. If we have two traits, then an EXAMPLE setting of this can be:
`parameter_signs=list(signsA=rbind(c("+","-"),c(0,"+")),`
`signsSyy=rbind(c(NA,0),c(0,NA))`. This means that `A` is upper triangular with positive values on the diagonal and a negative value on the off-diagonal, `Syy` is diagonal and `A` is also diagonal. It is advisable to set now `Atype="Any"` and `Syytype="Any"` (see further description).

If the given model parameter is to be estimated by a generalized least squares (currently `mPsi` and `vY0`), then the sign specifications are ignored. However, it is possible to set specific values. Furthermore, the package does not check (for `A` and `Syy`) if the specifications here agree with the `Atype`, `Syytype` and `diagA`. The settings in `signsA` and `signsSyy` will override the other settings. Hence, it is up to the user to make sure that the settings of `signsA` and `signsSyy` are consistent with `Atype`, `Syytype` and `diagA`. It is advisable to use `signsA` with `"+"` on the diagonal and have `diagA=NULL`. The diagonal of `Syy` is forced to be positive (unless `"-"` is used on the diagonal of `signsSyy` but this is strongly discouraged) so it is advisable to keep `NA` on the diagonal of `signsSyy` and not put there

"+" there. Hence, in particular using the signs mechanism result in a wrong class of the matrix (e.g. Atype="SymmetricPositiveDefinite", but after corrections for the provided entries in signsA one obtains a non-symmetric A with complex, negative-real-part eigenvalues). Lastly, using signsA and signsSyy can result in a wrong amount of dof and in turn incorrect AICc and BIC values. What the code does is subtracts the amount of fixed values in signsA and signsSyy from the amount of free parameters used to estimate A and Syy. For example if one sets Atype="SingleValueDiagonal" (estimated by one free parameter) but specified two off-diagonal values, then the amount of dofs from A will be -1!! The ONLY fail-safe way to use this is to set Atype="Any" (if signsA used) and Syytype="Any" (if signsSyy used). If using Syytype="Any" and signsSyy the it is strongly advisable to set the entries either below or above the diagonal ofSyy to 0. The reason is that $\Sigma_{yy}\Sigma_{yy}^T$ enters the likelihood and not the given value of Σ_{yy}. Hence, having values below (or respectively above) the diagonal results in an overparameterized model. The package has the option of mixing different matrix types with specifying values in it but this is only for advanced users who need to dig into the code to see what the dof should be and if it is possible to find a correspondence between the parametrization and settings. If entries of mPsi and vY0 are pre-specified, then the dof are correctly adjusted for this.

The found point is described by a list containing four fields. The first field HeuristicSearchPointFinalFind is the parametrization of the model parameters at the considered point with the value of the log-likelihood. The field ParamsInModel is the point estimate of the parameters of the SDE. The field ParamSummary are different composite (evaluated at the tree's height) and summary statistics, The field phylhalflife are the eigenvalues, eigenvectors and phylogenetic half lives associated with the A matrix, expmtA is $\exp(-A*(treeheight))$, mPsi.rotated is each of the regime effects multiplied by $(1-\exp(-A*(treeheight)))$, cov.matrix is the trait vector covariance matrix at the tree's height, corr.matrix is the trait vector correlation matrix at the tree's height, trait.regression is a list consisting of regression coefficients when taking each trait in turn and calculating its conditional expectation on all of the other trait, stationary.cov.matrix is the stationary covariance matrix of process if it exists (i.e. the eigenvalues have positive real part), stationary.corr.matrix is the stationary correlation matrix of process if it exists (i.e. the eigenvalues have positive real part), StS the infinitesimal covariance matrix $\Sigma_{yy}\Sigma_{yy}^T$, LogLik the log-likelihood, dof the degrees of freedom, m2loglik is -2log–likelihood, aic is the Akaike information criterion, aic.c is the Akaike information criterion corrected for small sample size, sic is the Schwarz information criterion, bic is the Bayesian information criterion (which is the same as the Schwarz information criterion) and RSS is the residual sum of squares. The field RSS_non_phylogenetic is a residual sum of squares calculated without correcting for the phylogeny–induced between species correlations, while the extension conditional_on_predictors indicates that we consider the RSS for the variables labelled as responses conditioned on the remaining variables. The R2_phylaverage field is R2, where the alternative model is the phylogenetically weighted sample average (see OU_phylreg).

From version 2.0.0 of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to mData.

From version 2.0.0 of **mvSLOUCH** the parameter calcCI has been removed. The package now offers the possibility of bootstrap confidence intervals, see function parametric.bootstrap.

Value

FinalFound	The point where the search procedure stopped. See Details for the description.
MaxLikFound	The point with the highest likelihood found by the search procedure, if it is the same as the final point then this field equals "Same as final found".

Warning

The estimation can take a long time and should be repeated a couple of times so that it is run from different starting positions. The function can produce (a lot of) warnings and errors during the search procedure, this is nothing to worry about.

Note

The **ouch** package considers a similar model and looking at it could be helpful.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Mitov, V. and Bartoszek, K. and Asimomitis, G. and Stadler, T. (2020) Fast likelihood calculation for multivariate Gaussian phylogenetic models with shifts Theoretical Population Biology 131:66-78.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

`PCMLik`, `hansen`, `SummarizeOUCH`, `simulOUCHProcPhylTree`, `parametric.bootstrap`, `optim`

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)
```

```
### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the OUOU model.
## 3D model
## OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
## A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
## Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
## 2D model used to reduce running time on CRAN
OUOUparameters<-list(vY0=matrix(c(1,-1),nrow=2,ncol=1),
A=rbind(c(9,0),c(0,5)),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

### Try to recover the parameters of the OUOU model.
### maxiter here set to minimal working possibility, in reality it should be larger
### e.g. default of c(10,100)
### Also the Atype and Syytype variables should be changed, here set as simplest
### for speed of evaluation, e.g. Atype="DecomposablePositive", Syytype="UpperTri"
OUOUestim<-ouchModel(phyltree,OUOUdata,regimes,Atype="SingleValueDiagonal",
Syytype="SingleValueDiagonal",diagA="Positive",maxiter=c(1,1))
RNGversion(as.character(getRversion()))

## Not run:  ##It takes too long to run this
### And finally bootstrap with particular interest in the evolutionary regression
OUOUbootstrap<-parametric.bootstrap(estimated.model=OUOUestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression"),regimes=regimes,root.regime="small",
M.error=NULL,predictors=c(2),kY=NULL,numboot=5,Atype=NULL,Syytype=NULL,diagA=NULL)

## End(Not run)
```

`OU_phylreg` *Performs a phylogenetic regression under a given OU model of evolution*

Description

The `OU_phylreg` function does a phylogenetic regression for given response and design matrices under a multivariate OU model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
OU_phylreg(mY, mD, phyltree, modelParams, regimes = NULL, kY = NULL, M.error = NULL,
signif_level = 0.05, regimes.times = NULL, root.regime = NULL, b_GLSB = FALSE,
b_GLSX0 = FALSE, signsB = NULL, signsvX0 = NULL, estimate.root.state = FALSE)
```

Arguments

`mY` A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

`mD` A design matrix with the rows corresponding to the traits in the tips species while the columns correspond to the unknown regression variables. The number or rows have to correspond to the number of elements in `mY`, the data are assumed to be stacked by species. If `NA` it is assumed to be the design matrix to estimate regression parameters under the given model of evolution, see Details. If it is `"phylaverage"`, then a phylogenetically weighted average is calculated, see Details.

`phyltree` The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

`modelParams` List of model parameters of the BM/OUOU/OUBM model as `ParamsInModel` part of output of `BrownianMotionModel/ouchModel/mvslouchModel`. Some of them can be `NA` in order to be estimated by the regression procedure, see Details.

`regimes` A vector or list of regimes. If vector then each entry corresponds to each of the branches of `phyltree`, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

`kY` Number of "Y" (response) variables if the considered model is an OUBM one. The first `kY` columns of `mY` are the "OU" ones, while the rest the "BM" ones. In more detail this value determines the number of columns of the `mY` matrix to treat as response variables ("OU" ones). For example, a value of 1 means that only the first column is treated as a response variable, while a value of 3 means the first three columns are treated as response variables. Any predictor variables ("BM" ones) the user is interested in setting for a particular model should therefore be placed in the final columns of the `mY` matrix, allowing for selecting select `kY` columns before this as response variables ("OU" ones).

`M.error` An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either

a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,

- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`signif_level` The significance level to be taken when calculating regression confidence intervals, i.e. $(1 - signif_level) \cdot 100$ percent confidence intervals are returned.

`regimes.times` A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

`root.regime` The regime at the root of the tree. If not given, then it is taken as the regime that is present on the daughter lineages stemming from the root and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

`b_GLSB` If the evolutionary model is an OUBM one (and `mD` is `NA`), then should the B matrix be also estimated. If B is completely provided in `modelParams`, then that value is taken as an initial guess for the regression estimation procedure (as the between-species-between-traits variance-covariance matrix depends on `B`).

`b_GLSX0` If the evolutionary model is an OUBM or BM one (and `mD` is `NA`), then should the X_0 ancestral vector also be estimated. If X_0 is completely provided in `modelParams`, then that value is taken as an initial guess for the regression estimation procedure (as in the OUBM the design matrix depends on `vX0`, in the BM case this value is ignored).

`signsB` A matrix of constraints on the estimation of `B`, with the same dimensions as `B`, if `b_GLBS` is `TRUE`. Inside this matrix the possible values are `"+"` if the given entry is to be positive, `"-"` if the given entry is to be negative, `x`, where `x` is a number, if the entry is to be set to specified value or `NA` if the entry is to be freely estimated. This option is still in an experimental stage so some setups might not work (please report).

`signsvX0` A matrix of constraints on the estimation of `vX0`, with the same dimensions as `vX0`, if `b_GLBX0` is `TRUE`. Inside this matrix the possible values are `"+"` if the given entry is to be positive, `"-"` if the given entry is to be negative, `x`, where `x` is a number, if the entry is to be set to specified value or `NA` if the entry is to be freely estimated. This option is still in an experimental stage so some setups might not work (please report).

`estimate.root.state`
Should the root state be estimate `TRUE` (not recommended) or set at the optimum `FALSE` (recommended). Root state estimation is usually unreliable hence if fossil measurements are available prediction based on them and the estimated model will probably be more accurate. If there is only one regime, then estimation of the root state separately is impossible and will not be allowed.

Details

The matrix algebra calculations are done using the likelihood function offerred by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

For a given input data matrix, mY, the function considers the stacking of it by rows (i.e. stacking species by species). Let Y = vec(mY), i.e. Y<-c(t(mY)), V be the between-species-between-traits variance-covariance matrix (under the parameters passed in modelParams). The function calculates the value of the generalized least squares estimator (not directly, but as a transformation of the likelihood provided by **PCMBase**)

$$v = (D^T V^{-1} D)^{-1} D^T V^{-1} Y.$$

The user can provide the design matrix directly or if mD is NA, then the design matrix induced by the evolutionary model in modelParams is assumed. The following parameters can be estimated: vX0 (if b_GLSX0 is TRUE, BM model); mPsi, vY0 (if estimate.root.state is TRUE, otherwise set at optimum) for OUOU model; vX0 (if b_GLSX0 is TRUE) mPsi, vY0 (if estimate.root.state is TRUE, otherwise set at optimum), B (if b_GLSB is TRUE). One can constrain (some of) the elements of the matrices to be estimated to be postive, negative or equal to some value. For B and vX0 this was described in the description of the arguments of signsB and signsvX0. For mPsi and vY0 one does this in the respective entries of modelParams. There matrix entries can be set to "+", "-", NA or some specific value. In the OUBM case the model specfic design matrix is not derived from the conditional expectation of all of the responses on all of the predictors, but from the conditional expectations of each tip species independently (as if V were block diagonal). This is as the joint condtional expecation design matrix cannot be calculated at the moment in an efficient manner and would cause a serious computational bottleneck. However this only makes a difference if B is to be estimated inside the GLS.

Special support is given if one wants to compute a phylogenetically weighted mean. If mD is set to "phylaverage", then it is calculated as

$$D_p = 1_n \otimes Id_k,$$

where 1_n is a column vector of n ones and Id_k is the identity matrix with rows and columns equalling the number or columns of mY.

Value

A list with the following entries

vGLSest The regression estimates

regression.covariance.matrix The covariance matrix between regression estimates.

regression.confidence.intervals The confidence intervals for each estimated parameter.

modelParams The model parameters updated if anything was estimated from them in the procedure.

mD The used or calculated design matrix.

RSS The residual sum of squares.

R2_average R2, where the alternative model is the sample average.

R2_phylaverage R2, where the alternative model is the phylogenetically weighted sample average, i.e. the design matrix is D_p.

RSS_average The RSS with respect to the sample average.

RSS_phylaverage The RSS with respect to the phylogenetically weighted sample average.

phyltree The phylogeny used, returned as in the estimation procedure some additional fields are calculated. This could help in a speed up if the `OU_phylreg` is used in some iterative procedure.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the OUOU model.
## 3D model
## OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
## A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
## Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
## 2D model used to reduce running time on CRAN
OUOUparameters<-list(vY0=matrix(c(1,-1),nrow=2,ncol=1),
A=rbind(c(9,0),c(0,5)),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
```

```
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

OUOUparameters_reg<-OUOUparameters
OUOUparameters_reg$mPsi<-apply(OUOUparameters_reg$mPsi,c(1,2),function(x){NA})
OUOUparameters_reg$vY0<-apply(OUOUparameters_reg$vY0,c(1,2),function(x){NA})
## estimate parameters under OUOU model
OU_phylreg(OUOUdata, NA, phyltree, OUOUparameters_reg, regimes=regimes,
kY=NULL, M.error=NULL)
```

OU_RSS *Calculates the RSS under a (multivariate) phylogenetic OU model of evolution*

Description

The OU_RSS function calculates the residual sum of squares (RSS) for given data under a multivariate OU model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
OU_RSS(mY, phyltree, modelParams, M.error = NULL, do_centre = NA,
regimes = NULL, regimes.times = NULL, root.regime = NULL)
```

Arguments

mY	A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field phyltree$tip.label), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices 1:n, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.
phyltree	The phylogeny in phylo format. The tree can be obtained from e.g. a nexus file by the read.nexus() function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices 1:n and the root index n+1. The root.edge field is ignored.
modelParams	List of model parameters of the BM/OUOU/OUBM model as ParamsInModel part of output of BrownianMotionModel/ouchModel/mvslouchModel.
M.error	An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities : • a single number that is a common measurement error for all tips and species, • a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,

- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`do_centre` Should the data, `mY`, be centred (admissable values are `"average"`, `"phylaverage"` or `"evolutionary_model"`) or not (`NA`). If `"average"`, then each column (trait) is centred by its arithmetic average, if `"evolutionary_model"`, then `mY` is centred by the expectation under the evolutoniary model (inferred from `modelParams`) and if `"phylaverage"`, then `mY` is centred by a phylogenetically weighted arithmetic average (see Details).

`regimes` A vector or list of regimes. If vector then each entry corresponds to each of the branches of `phyltree`, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

`regimes.times` A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

`root.regime` The regime at the root of the tree. If not given, then it is taken as the regime that is present on the daughter lineages stemming from the root and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

Details

The matrix algebra calculations are done using the likelihood function offerred by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

For a given input data matrix, `mY`, the function considers the stacking of it by rows (i.e. stacking species by species). Let Y = vec(`mY`), i.e. `Y<-c(t(mY))`, V be the between-species-between-traits variance-covariance matrix (under the parameters passed in `modelParams`) and `v` a centring vector (if `do_centre` is `NA`, then $v = 0$). The function calculates the value of the quadratic form

$$(Y - v)^T V^{-1}(Y - v).$$

A special centring is when `do_centre` equals `"phylaverage"`. In this situation the centring vector is a phylogenetically weighted average, i.e.

$$v = (D^T V^{-1} D)^{-1} D^T V^{-1} Y,$$

where denoting 1_n as a column vector of n ones and Id_k as the identity matrix with rows and columns equalling the number or columns of mY,

$$D = 1_n \otimes Id_k.$$

Value

The value of the residual sum of squares, quadratic form with respect to the between-species-between-traits precision matrix. Also the used phylogeny is returned.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the OUOU model.
## 3D model
## OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
## A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
## Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
## 2D model used to reduce running time on CRAN
OUOUparameters<-list(vY0=matrix(c(1,-1),nrow=2,ncol=1),
A=rbind(c(9,0),c(0,5)),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

## below will return the RSS under the assumed OUOU model of evolution
OU_RSS(OUOUdata, phyltree, OUOUparameters, M.error=NULL,
```

```
do_centre="evolutionary_model", regimes = regimes)
```

OU_xVz *Performs a vector matrix vector multiplcation under a (multivariate) phylogenetic OU model of evolution*

Description

The OU_xVz function performs a vector matrix vector multiplcation for given data under a multivariate OU model evolving on the phylogeny. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
OU_xVz(mX, mZ, phyltree, modelParams, M.error = NULL, do_centre = NA,
regimes = NULL, regimes.times = NULL, root.regime = NULL)
```

Arguments

mX
: The first data matrix for the vector matrix vector multiplication. A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field phyltree$tip.label), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices 1:n, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

mZ
: The second data matrix for the vector matrix vector multiplication. A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field phyltree$tip.label), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices 1:n, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

phyltree
: The phylogeny in phylo format. The tree can be obtained from e.g. a nexus file by the read.nexus() function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices 1:n and the root index n+1. The root.edge field is ignored.

modelParams
: List of model parameters of the BM/OUOU/OUBM model as ParamsInModel part of output of BrownianMotionModel/ouchModel/mvslouchModel. Same model is assumed for both mX and mZ.

M.error
: An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

  - a single number that is a common measurement error for all tips and species,

- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`do_centre` Should the data, `mX`, `mZ`, be centred (admissable values are `"average"`, `"phylaverage"` or `"evolutionary_model"`) or not (`NA`). If `"average"`, then each column (trait) is centred by its arithmetic average, if `"evolutionary_model"`, then `mX` and `mZ` are centred by the expectation under the evolutoniary model (inferred from `modelParams`) and if `"phylaverage"`, then `mX` and `mZ` are centred by phylogenetically weighted arithmetic averages (see Details).

`regimes` A vector or list of regimes. If vector then each entry corresponds to each of the branches of `phyltree`, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

`regimes.times` A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

`root.regime` The regime at the root of the tree. If not given, then it is taken as the regime that is present on the daughter lineages stemming from the root and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

Details

The matrix algebra calculations are done using the likelihood function offerred by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

For given input data matrices, `mX` and `mZ`, the function considers the stacking of them by rows (i.e. stacking species by species). Let X = vec(`mX`), i.e. `X<-c(t(mX))`, Z = vec(`mZ`), i.e. `Z<-c(t(mZ))`, V be the between-species-between-traits variance-covariance matrix (under the parameters passed in `modelParams`) and `vx`, `vz` be centring vectors (if `do_centre` is `NA`, then $vx = vz = 0$). The function calculates the value of the vector matrix vector multiplication

$$(X - vx)^T V^{-1} (Z - vz).$$

A special centring is when do_centre equals "phylaverage". In this situation the centring vector is a phylogenetically weighted average, i.e.

$$vx = (D^T V^{-1} D)^{-1} D^T V^{-1} X, vz = (D^T V^{-1} D)^{-1} D^T V^{-1} Z,$$

where denoting 1_n as a column vector of n ones and Id_k as the identity matrix with rows and columns equalling the number or columns of mY,

$$D = 1_n \otimes Id_k.$$

Value

The value of the vector matrix vector multiplication with respect to the between-species-between-traits precision matrix. Also the used phylogeny is returned.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the OUOU model.
## 3D model
## OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
## A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
## Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
## 2D model used to reduce running time on CRAN
OUOUparameters<-list(vY0=matrix(c(1,-1),nrow=2,ncol=1),
A=rbind(c(9,0),c(0,5)),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)))

### Now simulate the data.
```

```
OUOUdata1<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata1<-OUOUdata1[phyltree$tip.label,,drop=FALSE]

OUOUdata2<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata2<-OUOUdata2[phyltree$tip.label,,drop=FALSE]

OU_xVz(OUOUdata1, OUOUdata2, phyltree, OUOUparameters, M.error=NULL,
do_centre="evolutionary_model", regimes = regimes)
```

`parametric.bootstrap` *Parametric bootstrap for confidence intervals*

Description

The function performs a parametric bootstrap for confidence intervals for estimates of the evolutionary model. The user may specify what parameters are to have their confidence intervals returned. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
parametric.bootstrap(estimated.model, phyltree,
values.to.bootstrap = NULL, regimes = NULL,
root.regime = NULL, M.error = NULL, predictors = NULL,
kY = NULL, numboot = 100, Atype = NULL, Syytype = NULL,
diagA = NULL, parameter_signs = NULL, start_point_for_optim = NULL,
parscale = NULL, min_bl = 0.0003, maxiter = c(10,50,100), estimateBmethod="ML")
```

Arguments

`estimated.model`
: An estimated by evolutionary model. It can be e.g. the output of `BrownianMotionModel()`, `ouchModel()`, `mvslouchModel()` or `estimate.evolutionary.model()`. In the last case the model under `BestModel` is analyzed.

`phyltree`
: The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

`values.to.bootstrap`
: A vector of parameter/composite statistic names that the user is interested in. They are extracted from the bootstrapped elements for easy access.

`regimes`
: A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`. If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

root.regime
: The regime at the root of the tree. If not given, then it is taken as the regime that is present on the root's daughter lineages and is the most frequent one in the `regimes` vector. If more than one regime has the same maximum frequency, then alphabetically first one of the maximum ones is taken.

M.error
: An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version 2.0.0 of mvSLOUCH it is impossible to pass a single joint measurement error matrix for all the species and traits.

predictors
: A vector giving the numbers of the columns from the original data which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided then the "X" variables are treated as predictors, but this only for the OUBM models (for the others in this case none are treated as predictors).

kY
: Number of "Y" (response) variables, for the OUBM models. The first `kY` columns of `mY` are the "OU" ones, while the rest the "BM" ones. In more detail this value determines the number of columns of the (simulated) data matrix to treat as response variables ("OU" ones). For example, a value of 1 means that only the first column is treated as a response variable, while a value of 3 means the first three columns are treated as response variables. Any predictor variables ("BM" ones) the user is interested in setting for a particular model should therefore be placed in the final columns of the data matrix, allowing for selecting select `kY` columns before this as response variables ("OU" ones). If `NULL` then it is extracted from the provided model parameters in `estimated.model`.

numboot
: The number of bootstraps to perform.

Atype
: The class of the `A` matrix. It can take one of the following values: `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"SymmetricPositiveDefinite"`, `"Symmetric"`, `"DecomposablePositive"`, `"DecomposableNegative"`, `"DecomposableReal"`, `"Invertible"`, `"Any"`. If `NULL` then it is extracted from the provided model parameters in `estimated.model`.

Syytype
: The class of the Syy matrix, ignored if `evolmodel` equals `"BM"`. Otherwise it can take one of the following values: `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"Any"`. If `NULL` then it is extracted from the provided model parameters in `estimated.model`.

diagA
: Should the diagonal of A be forced to be positive ("Positive"), negative ("Negative") or the sign free to vary (NULL). However, setting this to a non-NULL value when evolmodel is "mvslouch" might be (but simulations concerning this are not conclusive) slightly detrimental to the optimization process if Atype is "DecomposablePositive", "DecomposableNegative", or "DecomposableReal". In these cases A is parametrized by its eigendecomposition. Additional exponentiation of the diagonal, to ensure positivity, could (but this is uncertain) make the exploration of the likelihood surface more difficult. In the case of Atype being "SymmetricPositiveDefinite", the diagonal is always guaranteed to be positive. If NULL then the function checks if it is not in the provided model parameters in estimated.model.

parameter_signs
: WARNING: ONLY use this option if you understand what you are doing! This option is still in an experimental stage so some setups might not work (please report). A list allowing the user to control whether specific entries for each model parameter should be positive, negative, zero or set to a specific (other) value. The entries of the list have to be named, the admissible names are "signsA" (for A matrix), "signsB" (for B matrix), "signsSyy" (for Syy matrix) and "signsmPsi" (for mPsi matrix) and "signsvY0" (for vY0 matrix). Any other entry in this list will be ignored. Each entry of the list has to be a matrix of appropriate size, i.e. of the size of the parameter to which it corresponds. Inside this matrix the possible values are "+" if the given entry is to be positive, "-" if the given entry is to be negative, x, where x is a number, if the entry is to be set to specified value or NA if the entry is to be freely estimated. See estimate.evolutionary.model, ouchModel and mvslouchModel for further details, examples and important warnings!

start_point_for_optim
: A named list with starting parameters for of the parameters for be optimized by optim(), currently only A and Syy for OUOU and OUBM models, i.e. will not work with BM model. One may provide both or only one of them. Make sure that the parameter is consistent with the other parameter restrictions as no check is done and this can result in undefined behaviour. For example one may provide this as (provided dimensions and other parameter restrictions agree)

```
start_point_for_optim=list(A=rbind(c(2,0),(0,4)),
Syy=rbind(c(1,0.5),c(0,2))).
```

This starting point is always jittered in each bootstrap replicate as the employed "Nelder-Mead" method in optim() is deterministic.

parscale
: A vector to calculate the parscale argument for optim. It is a named vector with 3 entries, e.g. c("parscale_A"=3,"logparscale_A"=5,"logparscale_other"=1). The entry parscale_A is the scale for entries of the A matrix, logparscale_A is the scale for entries of the A matrix that are optimized over on the logarithmic scale, e.g. if eigenvalues are assumed to be positive, then optimization is done over log(eigenvalue) for A's eigendecomposition and logparscale_other is the scale for entries other then of A that are done on the logarithmic scale (e.g. Syy's diagonal, or other entries indicated as positive via parameter_signs). If not provided (or if a name of the vector is misspelled), then made equal to the ex-

ample value provided above. For other elements, then mentioned above, that are optimized over by `optim()`, `1` is used for `optim()`'s `parscale`. It is advised that the user experiments with a couple of different values and reads `optim`'s man page.

`min_bl` Value to which **PCMBase**'s `PCMBase.Threshold.Skip.Singular` should be set. It indicates that branches of length shorter than `min_bl` should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. Furthermore, **mvSLOUCH** sets all branches in the tree shorter than `min_bl` to `min_bl`. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

`maxiter` The maximum number of iterations for different components of the estimation algorithm. A vector of three integers. The first is the number of iterations for phylogenetic GLS evaluations, i.e. conditional on the other parameters, the regime optima, perhaps `B`, and perhaps initial state are estimated by a phylogenetic GLS procedure. After this the other (except of `B` in OUBM model case) parameters are optimized over by `optim()`. This first entry controls the number of iterations of this procedure. The second is the number of iterations inside the iterated GLS for the OUBM model. In the first step regime optima and `B` (and perhaps initial state) are estimated conditional on the other parameters and current estimate of `B`, then the estimate of `B` is update and the same phylogenetic GLS is repeated (second entry of `maxiter` number of times). Finally, the third is the value of `maxiter` passed to `optim()`, apart from the optimization in the Brownian motion and measurement error case. If the bootstrapped model is a Brownian motion one, then this parameter is ignored, if OUOU, then the second entry is ignored.

`estimateBmethod`
Only relevant for OUBM models, should `B` be estimated by maximum likelihood (default value `"ML"`) or generalized least squares (value `"GLS"`).

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

The setting `Atype="Any"` means that one assumes the matrix `A` is eigendecomposable. If the estimation algorithm hits a defective `A`, then it sets the log-likelihood at the minimum value and will try to get out of this dip.

Value

A list with all the bootstrap simulations is returned. The elements of the list are the following.

`paramatric.bootstrap.estimation.replicates`
: A list of length equalling `numboot`. Each element is the result of the bootstrap replicate - the estimation results in the format of the output of **mvSLOUCH** functions, with an additional field `data`, the simulated data.

`bootstrapped.parameters`
: If `values.to.bootstrap` is not `NULL` then a list of length equalling length of `values.to.bootstrap`. Each element corresponds to the respective element of `values.to.bootstrap` and contains a list of the bootstrapped values of this element.

Warning

The estimation can take a long time and hence many bootstrap replicates will take even more time.The code can produce (a lot of) warnings and errors during the search procedure, this is nothing to worry about.

Note

The **ouch** package implements a parametric bootstrap and reading about it could be helpful.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

W. H. C. Kiang (2024) Exact Expressions for the Log-likelihood's Hessian in Multivariate Continuous-Time Continuous-Trait Gaussian Evolution along a Phylogeny, ArXiv e-prints:2405.07394.

See Also

`BrownianMotionModel`, `estimate.evolutionary.model`, `mvslouchModel`, `ouchModel`, `bootstrap`, `optim`

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

BMparameters<-list(vX0=matrix(0,nrow=3,ncol=1),
Sxx=rbind(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))

### Now simulate the data.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx)
BMdata<-BMdata[phyltree$tip.label,,drop=FALSE]

### Recover the parameters of the Brownian motion.
BMestim<-BrownianMotionModel(phyltree,BMdata)

### And finally obtain bootstrap confidence intervals for some parameters
BMbootstrap<-parametric.bootstrap(estimated.model=BMestim,phyltree=phyltree,
values.to.bootstrap=c("vX0","StS"),M.error=NULL,numboot=2)
RNGversion(as.character(getRversion()))

## Not run: ##It takes too long to run this
### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-list(vY0=matrix(c(1,-1),ncol=1,nrow=2),A=rbind(c(9,0),c(0,5)),
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[phyltree$tip.label,,drop=FALSE]

### Try to recover the parameters of the mvOUBM model.
OUBMestim<-mvslouchModel(phyltree,OUBMdata,2,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive",maxiter=c(10,50,100))

### And finally bootstrap with particular interest in the evolutionary and optimal
### regressions

OUBMbootstrap<-parametric.bootstrap(estimated.model=OUBMestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression","optimal.regression"),
regimes=regimes,root.regime="small",M.error=NULL,predictors=c(3),kY=2,
numboot=5,Atype="DecomposablePositive",Syytype="UpperTri",diagA="Positive",
maxiter=c(10,50,100))
```

```
### We now demonstrate an alternative setup
### Define SDE parameters to be able to simulate data under the OUOU model.
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),
Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

### Try to recover the parameters of the OUOU model.
estimResults<-estimate.evolutionary.model(phyltree,OUOUdata,regimes=regimes,
root.regime="small",M.error=NULL,repeats=3,model.setups=NULL,predictors=c(3),kY=2,
doPrint=TRUE,pESS=NULL,maxiter=c(10,50,100))

### And finally bootstrap with particular interest in the evolutionary regression
OUOUbootstrap<-parametric.bootstrap(estimated.model=estimResults,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression"),
regimes=regimes,root.regime="small",M.error=NULL,predictors=c(3),kY=NULL,
numboot=5,Atype=NULL,Syytype=NULL,diagA=NULL)

## End(Not run)
```

---

phyltree_paths *Extract path information from a phylogenetic tree*

Description

The function computes for each node its path to the root and its distance to the root. It returns an “enhanced” phylo type tree.

Usage

```
phyltree_paths(phyltree)
```

Arguments

phyltree	The phylogeny - an object of class phylo, i.e. tree in **ape** format. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices 1:n and the root index n+1.

Details

The function removes a root edge, i.e. $root.edge if one is present.

Value

The function returns a phylo type tree with the below additional fields.

- `Ntips` Number of tips on the tree.
- `path.from.root` A list of length equalling the number of nodes. Each entry is a list made up of two fields `nodes` and `edges`. `nodes` are the nodes on the path to the root and `edges` the edges.
- `time.of.nodes` A vector of length equalling the number of nodes. Each entry is the node's distance from the root. This is only calculated if the input tree has the `$edge.length` field.
- `tree_height` The height of the tree if it is ultrametric, otherwise the length of the longest path from root to tip.
- `tip_species_index` The node numbers corresponding to tip nodes, should equal `1:n`.
- `internal_nodes_index` The node numbers corresponding to internal nodes, should equal `(n+1):(2n-1)`.
- `root_index` The node number corresponding to the root, should equal `n+1`.

Note

The `ape` and `phangorn` packages include related tree manipulation functions.

Author(s)

Krzysztof Bartoszek

See Also

`ape`, `phangorn`

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
phyltree<-ape::rtree(5)
phyltree_augmented<-phyltree_paths(phyltree)
RNGversion(as.character(getRversion()))
```

`plot.clustered_phylo` *Plots a clustered_phylo object.*

Description

The function plots a clustered_phylo object allowing the user to differently visualize the different clades/clusters on the phylogeny and also the joining them subtree.

Usage

```
## S3 method for class 'clustered_phylo'
plot(x, clust_cols = NULL, clust_edge.width = NULL,
clust_edge.lty = NULL, clust_tip.color = "black", joiningphylo_col = "black",
joiningphylo_edge.width = 1, joiningphylo_edge.lty = 1, ...)
```

Arguments

- `x` A phylogenetic tree of class `clustered_phylo`, i.e. output of `mvSLOUCH::simulate_clustered_phylog`
- `clust_cols` Vector of colours of edges inside each cluster. Default `NULL`, corresponding to `"black"`. If length of this vector does not equal to the number of clusters, then it is recycled.
- `clust_edge.width` Numeric vector of widths of edges inside each cluster. Default `NULL`, corresponding to `1`. If length of this vector does not equal to the number of clusters, then it is recycled.
- `clust_edge.lty` Vector of an edge's type inside each cluster. Default `NULL`, corresponding to `1`. If length of this vector does not equal to the number of clusters, then it is recycled.
- `clust_tip.color` Vector of colours of tips' labels inside each cluster. Default `NULL`, corresponding to `"black"`. If length of this vector does not equal to the number of clusters, then it is recycled.
- `joiningphylo_col` Colour of edges inside the subtree joining the clusters.
- `joiningphylo_edge.width` Width of edges inside the subtree joining the clusters.
- `joiningphylo_edge.lty` Edges' type inside the subtree joining the clusters.
- `...` Other parameters to be passed to `plot.phylo()`. Notice that here we cannot have `edge.color`, `edge.width` and `edge.lty`.

Value

Same as `plot.phylo()`.

Author(s)

Krzysztof Bartoszek

References

Bartoszek K. and Vasterlund A. (2020) "Old Techniques for New Times": the RMaCzek package for producing Czekanowski's diagrams Biometrical Letters 57(2):89-118.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")

phyltree<-simulate_clustered_phylogeny(v_sizeclusts=c(5,5,5),f_simclustphyl="sim.bd.taxa_Yule1",
b_change_joining_branches=TRUE, joining_branchlengths=c(20,NA),joining="sim.bd.taxa_Yule1")

plot(phyltree,clust_cols=c("red","green","blue"),clust_edge.width=3,clust_edge.lty=c(1,2,3),
clust_tip.color=c("red","blue","green"),joiningphylo_col="black",joiningphylo_edge.width=3,
joiningphylo_edge.lty=1)

## and not plot without tip labels
plot(phyltree,clust_cols=c("red","green","blue"),clust_edge.width=3,clust_edge.lty=c(1,2,3),
joiningphylo_col="black",joiningphylo_edge.width=3,joiningphylo_edge.lty=1,show.tip.label=FALSE)

RNGversion(as.character(getRversion()))
```

---

simulate_clustered_phylogeny

Simulate a phylogenetic tree with a specified number of clades.

Description

Simulate a phylogenetic tree that has a given number of clades, each with a given number of tips.

Usage

```
simulate_clustered_phylogeny(v_sizeclusts, joining_branchlengths = NULL,
f_simclustphyl = "sim.bd.taxa_Yule1", joiningphyl = NULL,
b_change_joining_branches = FALSE, ...)
```

Arguments

`v_sizeclusts` A vector with the sizes of the clades/clusters.

`joining_branchlengths`
Default `NULL`, if `joiningphyl` is `NULL`, then has to be provided. A vector of two numbers. The first element are the lengths of the branches of the cluster joining phylogeny leading to the clusters. The second element will be the lengths of the "internal" branches of the cluster joining phylogeny. If only a single number is provided, then all the branches of the joining phylogeny will have their lengths equal to this value.

`f_simclustphyl` What function to use to simulate the phylogeny inside each cluster. The default value of "sim.bd.taxa_Yule1" corresponds to a pure birth tree generated by `ape::rphylo(n=clade_size,birth=1,death=0)`, without a root branch otherwise the user should pass an object of class `function` and its parameters in place of the `...` . The first parameter must be the number of contemporary leaves and be called `n`. The function has to return a valid `phylo` object.

joiningphyl
: By what phylogeny are the clades to be joined by. Either NULL (default), a phylo object, the character string "sim.bd.taxa_Yule1" or an object of class function. If NULL, then they are joined by a caterpillar (comb/pectinate) phylogeny with the branch lengths as provided by the joining_branchlengths parameter. If it is a phylo object, then they will be joined by it. Importantly the number of tips of this phylogeny has to equal the number of clusters. If "sim.bd.taxa_Yule1", then the joining phylogeny is simulated as a pure birth tree with tips equalling the number of clusters by ape::rphylo(). If it is a function, then this is used and its parameters are passed through .... The first parameter must be the number of contemporary leaves and be called n. The function has to return a valid phylo object.

b_change_joining_branches
: Logical, if joining phylogeny (parameter joiningphyl) was provided or simulated, should its branches be changed according to what was provided in joining_branchlengths (if it was not NULL). By default FALSE and the branch lengths are not changed.

...
: Parameters to be passed to user provided f_simclustphyl and joiningphyl functions. Unless one knows exactly what one is doing they should be passed by name. If there is a conflict of names, then one should pass wrapper functions around these functions where the names conflict is resolved.

Value

The resulting object is a clustered_phylo object which inherits from the phylo class and enhances it. Apart from the standard phylo fields it has two additional ones:

edges_clusters a named list with length equalling the number of clades/clusters plus 1. The first element of the list is called joining_tree and contains the indices (row numbers of the edge matrix, indices of the edge_length vector) of the edges inside the subtree joining the clusters. Afterwords element (i+1) is named cluster_i and contains a numeric vector with the indices of the edges inside clade i.

tips_clusters a named list with length equalling the number of clades/clusters. Each field of the list is a numeric vector containing the indices of the tips inside the clade. The names of element i of the list is cluster_i.

Author(s)

Krzysztof Bartoszek

References

Bartoszek K. and Vasterlund A. (2020) "Old Techniques for New Times": the RMaCzek package for producing Czekanowski's diagrams Biometrical Letters 57(2):89-118.

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
## We use a wrapper function for illustration
## a single phylo object
```

```
my_sim.bd.taxa<-function(n,...){
    ape::rphylo(n=n,...)
}
phyltree1<-simulate_clustered_phylogeny(v_sizeclusts=c(5,5,5),f_simclustphyl=my_sim.bd.taxa,
b_change_joining_branches=TRUE, joining_branchlengths=c(20,NA),joining=my_sim.bd.taxa,
birth=1,death=0)

RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
## The below code should return the same tree as above
phyltree2<-simulate_clustered_phylogeny(v_sizeclusts=c(5,5,5),f_simclustphyl="sim.bd.taxa_Yule1",
b_change_joining_branches=TRUE, joining_branchlengths=c(20,NA),joining="sim.bd.taxa_Yule1")

## The resulting phylogeny is not ultrametric, if ultrametricity is required, then some procedure
## has to be employed, e.g.
## phyltree1_u<-phytools::force.ultrametric(phyltree1, method="extend")
RNGversion(as.character(getRversion()))
```

`simulBMProcPhylTree` *Simulate data on a phylogeny under a (multivariate) Brownian motion model*

Description

Simulate data on a phylogeny under a (multivariate) Brownian motion model

Usage

```
simulBMProcPhylTree(phyltree, X0, Sigma, dropInternal = TRUE, M.error=NULL,
fullTrajectory=FALSE, jumpsetup=NULL, keep_tree = FALSE, step=NULL)
```

Arguments

`phyltree` The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

`X0` The ancestral, root state.

`Sigma` The diffusion matrix of the Brownian motion.

`dropInternal` Logical whether the simulated values at the internal nodes should be dropped.

`M.error` An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,

- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the phyltree object,
- NULL no measurement error.

From version 2.0.0 of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

fullTrajectory Should the full realization of the process or only node and tip values be returned

jumpsetup Either NULL or list describing the jump at speciation. In the second case:

jumptype In what way does the jump take place. Possible values are "ForBoth" the jump occurs at speciation and is common to both daughter lineages, "RandomLineage" the jump occurs just after speciation affecting exactly one daughter lineage, both descending branches have the same chance of being affected, "JumpWithProb" the jump occurs with probability jumpprob just after speciation independently on each daughter lineage independently.

jumpprob A value in $[0, 1]$ indicating the probability of a jump taking place, only matters if jumptype is "JumpWithProb" or "JumpWithProb".

jumpdistrib The distribution of the jump, currently only can take value "Normal".

vMean The expected value of the jump, a vector of appropriate length if the trait is multivariate.

mCov The variance of the jump, a matrix of appropriate dimensions if the trait is multivariate.

keep_tree Logical whether the used tree should be saved inside the output object. Useful for any future reference, but as the tree is enhanced for **mvSLOUCH**'s needs the resulting output object may be very large (it the number of tips is large).

step The step size of the simulation.

Value

If fullTrajectory is FALSE then returns a matrix with each row corresponding to a tree node and each column to a trait. Otherwise returns a more complex object describing the full realization of the process on the tree. If dropInternal is TRUE, then the entries for the internal nodes are changed to NAs. The ordering of the rows corresponds to the order of the nodes (their indices) in the phylo object. Hence, the first n rows will be the tip rows (by common phylo convention).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. (2014) Quantifying the effects of anagenetic and cladogenetic evolution. Mathematical Biosciences 254:42-57.

Bartoszek, K. (2016) A Central Limit Theorem for punctuated equilibrium. arXiv:1602.05189.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Felsenstein, J. (1985) Phylogenies and the comparative method. American Naturalist 125:1-15.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

BrownianMotionModel, SummarizeBM

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define Brownian motion parameters to be able to simulate data
### under the Brownian motion model.
BMparameters<-list(vX0=matrix(0,nrow=3,ncol=1),
Sxx=rbind(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))

### Now simulate the data.
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpdistrib="Normal",
vMean=rep(0,3),mCov=diag(1,3,3))
```

```
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx,
jumpsetup=jumpobj)
RNGversion(as.character(getRversion()))
```

---

simulMVSLOUCHProcPhylTree *Simulate data on a phylogeny under a (multivariate) OUBM model*

Description

Simulate data on a phylogeny under a (multivariate) OUBM model

Usage

```
simulMVSLOUCHProcPhylTree(phyltree, modelParams, regimes = NULL,
regimes.times = NULL, dropInternal = TRUE, M.error=NULL, fullTrajectory=FALSE,
jumpsetup=NULL,keep_tree=FALSE, step=NULL)
```

Arguments

phyltree — The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

modelParams — List of model parameters of mvOUBM model as `ParamsInModel` part of output of `mvslouchModel`.

regimes — A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`.If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

regimes.times — A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.

dropInternal — Logical whether the simulated values at the internal nodes should be dropped.

M.error — An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognizes the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,

- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the phyltree object,
- NULL no measurement error.

From version 2.0.0 of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

fullTrajectory should the full realization of the process or only node and tip values be returned

jumpsetup Either NULL or list describing the jump at speciation. In the second case:

jumptype In what way does the jump take place. Possible values are "ForBoth" the jump occurs at speciation and is common to both daughter lineages, "RandomLineage" the jump occurs just after speciation affecting exactly one daughter lineage, both descending branches have the same chance of being affected, "JumpWithProb" the jump occurs with probability jumpprob just after speciation independently on each daughter lineage independently.

jumpprob A value in $[0, 1]$ indicating the probability of a jump taking place, only matters if jumptype is "JumpWithProb" or "JumpWithProb".

jumpdistrib The distribution of the jump, currently only can take value "Normal".

vMean The expected value of the jump, a vector of appropriate length if the trait is multivariate.

mCov The variance of the jump, a matrix of appropriate dimensions if the trait is multivariate.

keep_tree Logical whether the used tree should be saved inside the output object. Useful for any future reference, but as the tree is enhanced for **mvSLOUCH**'s needs the resulting output object may be very large (it the number of tips is large).

step The step size of the simulation.

Value

If fullTrajectory is FALSE then returns a matrix with each row corresponding to a tree node and each column to a trait. Otherwise returns a more complex object describing the full realization of the process on the tree. If dropInternal is TRUE, then the entries for the internal nodes are changed to NAs. The ordering of the rows corresponds to the order of the nodes (their indices) in the phylo object. Hence, the first n rows will be the tip rows (by common phylo convention).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. (2014) Quantifying the effects of anagenetic and cladogenetic evolution. Mathematical Biosciences 254:42-57.

Bartoszek, K. (2016) A Central Limit Theorem for punctuated equilibrium. arXiv:1602.05189.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Hansen, T.F. and Pienaar, J. and Orzack, S.H. (2008) A comparative method for studying adaptation to randomly evolving environment. Evolution 62:1965-1977.

Labra, A., Pienaar, J. & Hansen, T.F. (2009) Evolution of thermophysiology in Liolaemus lizards: adaptation, phylogenetic inertia and niche tracking. The American Naturalist 174:204-220.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

mvslouchModel, SummarizeMVSLOUCH

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")
```

```
### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-list(vY0=matrix(c(1,-1),ncol=1,nrow=2),A=rbind(c(9,0),c(0,5)),
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data.
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpdistrib="Normal",
vMean=rep(0,3),mCov=diag(1,3,3))
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL,
jumpsetup=jumpobj)
RNGversion(as.character(getRversion()))
```

`simulOUCHProcPhylTree` *Simulate data on a phylogeny under a (multivariate) OU model*

Description

Simulate data on a phylogeny under a (multivariate) OU model

Usage

```
simulOUCHProcPhylTree(phyltree, modelParams, regimes = NULL,
regimes.times = NULL, dropInternal = TRUE, M.error=NULL, fullTrajectory=FALSE,
jumpsetup=NULL,keep_tree=FALSE,step=NULL)
```

Arguments

`phyltree`	The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.
`modelParams`	List of model parameters of OUOU model as `ParamsInModel` part of output of `ouchModel`.
`regimes`	A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`.If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.
`regimes.times`	A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL` then each branch is considered to be a regime.
`dropInternal`	Logical whether the simulated values at the internal nodes should be dropped.
`M.error`	An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

`fullTrajectory` should the full realization of the process or only node and tip values be returned

`jumpsetup` Either `NULL` or list describing the jump at speciation. In the second case:

`jumptype` In what way does the jump take place. Possible values are `"ForBoth"` the jump occurs at speciation and is common to both daughter lineages, `"RandomLineage"` the jump occurs just after speciation affecting exactly one daughter lineage, both descending branches have the same chance of being affected, `"JumpWithProb"` the jump occurs with probability `jumpprob` just after speciation independently on each daughter lineage independently.

`jumpprob` A value in $[0, 1]$ indicating the probability of a jump taking place, only matters if `jumptype` is `"JumpWithProb"` or `"JumpWithProb"`.

`jumpdistrib` The distribution of the jump, currently only can take value `"Normal"`.

`vMean` The expected value of the jump, a vector of appropriate length if the trait is multivariate.

`mCov` The variance of the jump, a matrix of appropriate dimensions if the trait is multivariate.

`keep_tree` Logical whether the used tree should be saved inside the output object. Useful for any future reference, but as the tree is enhanced for **mvSLOUCH**'s needs the resulting output object may be very large (it the number of tips is large).

`step` The step size of the simulation.

Value

If `fullTrajectory` is `FALSE` then returns a matrix with each row corresponding to a tree node and each column to a trait. Otherwise returns a more complex object describing the full realization of the process on the tree. If `dropInternal` is `TRUE`, then the entries for the internal nodes are changed to `NA`s. The ordering of the rows corresponds to the order of the nodes (their indices) in the `phylo` object. Hence, the first `n` rows will be the tip rows (by common `phylo` convention).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. (2014) Quantifying the effects of anagenetic and cladogenetic evolution. Mathematical Biosciences 254:42-57.

Bartoszek, K. (2016) A Central Limit Theorem for punctuated equilibrium. arXiv:1602.05189.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2023) Model Selection Performance in Phylogenetic Comparative Methods Under Multivariate Ornstein-Uhlenbeck Models of Trait Evolution, Systematic Biology 72(2):275-293.

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

hansen, ouchModel, simulOUCHProcPhylTree

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")
```

```
### Define SDE parameters to be able to simulate data under the OUOU model.
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
"large"=c(-1,1,0.5)),Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data.
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpdistrib="Normal",
vMean=rep(0,3),mCov=diag(1,3,3))
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL,jumpsetup=jumpobj)
RNGversion(as.character(getRversion()))
```

SummarizeBM — *Summarize parameters estimated under a Brownian motion model*

Description

Compiles a summary (appropriate moments, conditional moments, information criteria) of parameters of a Brownian motion model at a given time point. The user is recommended to install suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
SummarizeBM(phyltree, mData, modelParams, t = c(1), dof = NULL, M.error = NULL,
predictors = NULL, min_bl = 0.0003)
```

Arguments

phyltree	The phylogeny in phylo format. The tree can be obtained from e.g. a nexus file by the read.nexus() function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices 1:n and the root index n+1. The root.edge field is ignored.
mData	A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field phyltree$tip.label), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices 1:n, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.
modelParams	A list of model parameters, as returned in ParamsInModel part of BrownianMotionModel's output.
t	A vector of time points at which the summary is to be calculated. This allows for one to study (and plot) the (conditional) mean and covariance as functions of time. The function additionally returns the parameter summary at the tree's height.
dof	Number of unknown parameters in the model, can be extracted from the output of BrownianMotionModel(). If not provided all parameters are assumed unknown.

M.error — An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

predictors — A vector giving the numbers of the columns from `data` which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided, then none will be treated as predictors.

min_bl — Value to which **PCMBase**'s `PCMBase.Threshold.Skip.Singular` should be set. It indicates that branches of length shorter than `min_bl` should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

The `phyltree_paths()` function enhances the tree for usage by `mvSLOUCH`. Hence, to save time, it is advisable to first do `phyltree<-mvSLOUCH::phyltree_paths(phyltree)` and only then use it with `BrownianMotionModel()`.

From version `2.0.0` of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to `mData`.

From version `2.0.0` of **mvSLOUCH** the parameter `calcCI` has been removed. The package now offers the possibility of bootstrap confidence intervals, see function `parametric.bootstrap`.

Value

A list for each provided time point. See the help of BrownianMotionModel for what the summary at each time point is.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Analytical advances alleviate model misspecification in non-Brownian multivariate comparative methods, Evolution 78(3):389-400.

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Felsenstein, J. (1985) Phylogenies and the comparative method. American Naturalist 125:1-15.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

BrownianMotionModel, simulBMProcPhylTree, parametric.bootstrap

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define Brownian motion parameters to be able to simulate data
### under the Brownian motion model.
BMparameters<-list(vX0=matrix(0,nrow=3,ncol=1),
```

```
Sxx=rbind(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))

### Now simulate the data.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx)
BMdata<-BMdata[phyltree$tip.label,,drop=FALSE]

### Recover the parameters of the Brownian motion.
BMestim<-BrownianMotionModel(phyltree,BMdata)

### Summarize them.
BM.summary<-SummarizeBM(phyltree,BMdata,BMestim$ParamsInModel,t=c(1),
dof=BMestim$ParamSummary$dof)
RNGversion(as.character(getRversion()))
#\dontrun
{ ##It takes too long to run this
### Now obtain bootstrap confidence intervals for some parameters.
BMbootstrap<-parametric.bootstrap(estimated.model=BMestim,phyltree=phyltree,
values.to.bootstrap=c("vX0","StS"),,M.error=NULL,numboot=5)
}
```

SummarizeMVSLOUCH *Summarize parameters estimated under a multivariate OUBM motion model*

Description

Compiles a summary (appropriate moments, conditional moments, information criteria) of parameters of a multivariate OUBM model at a given time point. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
SummarizeMVSLOUCH(phyltree, mData, modelParams, regimes = NULL,
regimes.times = NULL, t = c(1), dof = NULL, M.error = NULL, predictors = NULL,
Atype = "Invertible", Syytype = "UpperTri", min_bl = 0.0003, maxiter = 50)
```

Arguments

phyltree — The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

mData — A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

modelParams
: A list of model parameters, as returned in `ParamsInModel` part of `mvslouchModel`'s output.

regimes
: A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`.If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

regimes.times
: A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL`, then each branch is considered to be constant a regime.

t
: A vector of time points at which the summary is to be calculated. This allows for one to study (and plot) the (conditional) mean and covariance as functions of time. The function additionally returns the parameter summary at the tree's height.

dof
: Number of unknown parameters in the model, can be extracted from the output of `mvslouchModel()`. If not provided all parameters are assumed unknown.

M.error
: An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

- a single number that is a common measurement error for all tips and species,
- a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
- a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
- a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
- NULL no measurement error.

From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

predictors
: A vector giving the numbers of the columns from `data` which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided then columns `(kY+1):ncol(mData)`, i.e. the "BM" ones, are treated as predictors.

Atype
: What class does the A matrix in the multivariate OUBM model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `Symmetric`, `SymmetricPositiveDefinite`, `"DecomposablePositive"`, `"DecomposableNegative"`, `"DecomposableReal"`, `"Invertible"`, `"TwoByTwo"`, `"Any"`

Syytype
: What class does the Syy matrix in the multivariate OUBM model belong to, possible values : `"SingleValueDiagonal"`, `"Diagonal"`, `"UpperTri"`, `"LowerTri"`, `"Symmetric"`, `"Any"`

min_bl — Value to which **PCMBase**'s PCMBase.Threshold.Skip.Singular should be set. It indicates that branches of length shorter than min_bl should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

maxiter — The maximum number of iterations inside the GLS estimation procedure. In the first step regime optima and B (and perhaps initial state) are estimated conditional on the other parameters and current estimate of B, then the estimate of B is update and the same phylogenetic GLS is repeated.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

The setting Atype="Any" means that one assumes the matrix A is eigendecomposable. If A is defective, then the output will be erroneous.

From version 2.0.0 of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to mData.

From version 2.0.0 of **mvSLOUCH** the parameter calcCI has been removed. The package now offers the possibility of bootstrap confidence intervals, see function parametric.bootstrap.

Value

A list for each provided time point. See the help of mvslouchModel for what the summary at each time point is.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Hansen, T.F. and Pienaar, J. and Orzack, S.H. (2008) A comparative method for studying adaptation to randomly evolving environment. Evolution 62:1965-1977.

Labra, A., Pienaar, J. & Hansen, T.F. (2009) Evolution of thermophysiology in Liolaemus lizards: adaptation, phylogenetic inertia and niche tracking. The American Naturalist 174:204-220.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

`slouch::model.fit`, `mvslouchModel`, `simulMVSLOUCHProcPhylTree`, `parametric.bootstrap`

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(3)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small")

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-list(vY0=matrix(1,ncol=1,nrow=1),A=matrix(0.5,ncol=1,nrow=1),
B=matrix(c(2),ncol=1,nrow=1),mPsi=cbind("small"=1,"large"=-1),
Syy=matrix(2,ncol=1,nrow=1),vX0=matrix(0,ncol=1,nrow=1),Sxx=diag(1,1,1),
Syx=matrix(0,ncol=1,nrow=1),Sxy=matrix(0,ncol=1,nrow=1))

### Now simulate the data.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[phyltree$tip.label,,drop=FALSE]

## Here we do not do any recovery step
OUBM.summary<-SummarizeMVSLOUCH(phyltree,OUBMdata,OUBMparameters,
regimes,t=c(1),dof=7,maxiter=2)

RNGversion(as.character(getRversion()))
## Not run:  ##It takes too long to run this
## now less trivial simulation setup
phyltree<-ape::rtree(5)
```

```
## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-list(vY0=matrix(c(1,-1),ncol=1,nrow=2),A=rbind(c(9,0),c(0,5)),
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syy=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[phyltree$tip.label,]

### Try to recover the parameters of the mvOUBM model.
OUBMestim<-mvslouchModel(phyltree,OUBMdata,2,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive")

### Summarize them.
OUBM.summary<-SummarizeMVSLOUCH(phyltree,OUBMdata,OUBMestim$FinalFound$ParamsInModel,
regimes,t=c(phyltree$tree_height),dof=OUBMestim$FinalFound$ParamSummary$dof,maxiter=50)

### And finally bootstrap with particular interest in the evolutionary and optimal
### regressions
OUBMbootstrap<-parametric.bootstrap(estimated.model=OUBMestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression","optimal.regression"),
regimes=regimes,root.regime="small",M.error=NULL,predictors=c(3),kY=2,
numboot=5,Atype="DecomposablePositive",Syytype="UpperTri",diagA="Positive")


## End(Not run)
```

SummarizeOUCH *Summarize parameters estimated under a (multivariate) OU motion model*

Description

Compiles a summary (appropriate moments, conditional moments, information criteria) of parameters of a (multivariate) OU model at a given time point. The user is recommended to install the suggested package **PCMBaseCpp** which significantly speeds up the calculations (see Details).

Usage

```
SummarizeOUCH(phyltree, mData, modelParams, regimes = NULL,
regimes.times = NULL, t = c(1), dof = NULL, M.error = NULL,
predictors = NULL, Atype = "Invertible", Syytype = "UpperTri", min_bl = 0.0003)
```

Arguments

phyltree
: The phylogeny in `phylo` format. The tree can be obtained from e.g. a `nexus` file by the `read.nexus()` function from the **ape** package. The "standard" **ape** node indexing is assumed: for a tree with n tips, the tips should have indices `1:n` and the root index `n+1`. The `root.edge` field is ignored.

mData
: A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be named by species (field `phyltree$tip.label`), if not, then a warning is thrown and the order of the species is assumed to be the same as the order in which the species are in the phylogeny (i.e. correspond to the node indices `1:n`, where n is the number of tips). The columns should be named by traits, otherwise a warning is thrown and generic names are generated.

modelParams
: A list of model parameters, as returned in `ParamsInModel` part of `ouchModel`'s output.

regimes
: A vector or list of regimes. If vector then each entry corresponds to each of `phyltree`'s branches, i.e. to each row of `phyltree$edge`.If list then each list entry corresponds to a tip node and is a vector for regimes on that lineage. If `NULL`, then a constant regime is assumed on the whole tree.

regimes.times
: A list of vectors for each tree node, it starts with 0 and ends with the current time of the species. In between are the times where the regimes (niches) changed. If `NULL`, then each branch is considered to be a regime.

t
: A vector of time points at which the summary is to be calculated. This allows for one to study (and plot) the (conditional) mean and covariance as functions of time. The function additionally returns the parameter summary at the tree's height.

dof
: Number of unknown parameters in the model, can be extracted from the output of `ouchModel()`. If not provided all parameters are assumed unknown.

M.error
: An optional measurement error covariance structure. The measurement errors between species are assumed independent. The program tries to recognize the structure of the passed matrix and accepts the following possibilities :

  - a single number that is a common measurement error for all tips and species,
  - a m element vector with each value corresponding to a variable, measurement errors are independent between variables and each species is assumed to have the same measurement errors,
  - a m x m ((number of variables) x (number of variables)) matrix, all species will have the same measurement error,
  - a list of length n (number of species), each list element is the covariance structure for the appropriate (numbering according to tree) species, either a single number (each variable has same variance), vector (of length m for each variable), or m x m matrix, the order of the list has to correspond to the order of the nodes in the `phyltree` object,
  - NULL no measurement error.

  From version `2.0.0` of **mvSLOUCH** it is impossible to pass a single joint measurement error matrix for all the species and traits.

predictors — A vector giving the numbers of the columns from data which are to be considered predictor ones, *i.e.* conditioned on in the program output. If not provided, then none will be treated as predictors.

Atype — What class does the A matrix in the multivariate OUBM model belong to, possible values : "SingleValueDiagonal", "Diagonal", "UpperTri", "LowerTri", Symmetric, SymmetricPositiveDefinite, "DecomposablePositive", "DecomposableNegative", "DecomposableReal", "Invertible", "TwoByTwo", "Any"

Syytype — What class does the Syy matrix in the multivariate OUBM model belong to, possible values : "SingleValueDiagonal", "Diagonal", "UpperTri", "LowerTri", "Symmetric", "Any"

min_bl — Value to which **PCMBase**'s PCMBase.Threshold.Skip.Singular should be set. It indicates that branches of length shorter than min_bl should be skipped in likelihood calculations. Short branches can result in singular covariance matrices for the transition density along a branch. The user should adjust this value if a lot of warnings are raised by **PCMBase** about singularities during the likelihood calculations. However, this does not concern tip branches-these cannot be skipped and hence should be long enough so that numerical issues are not raised.

Details

The likelihood calculations are done by the **PCMBase** package. However, there is a C++ backend, **PCMBaseCpp**. If it is not available, then the likelihood is calculated slower using pure R. However, with the calculations in C++ up to a 100-fold increase in speed is possible (more realistically 10-20 times). The **PCMBaseCpp** package is available from https://github.com/venelin/PCMBaseCpp.

The setting Atype="Any" means that one assumes the matrix A is eigendecomposable. If A is defective, then the output will be erroneous.

From version 2.0.0 of **mvSLOUCH** the data has to be passed as a matrix. To underline this the data parameter's name has been changed to mData.

From version 2.0.0 of **mvSLOUCH** the parameter calcCI has been removed. The package now offers the possibility of bootstrap confidence intervals, see function parametric.bootstrap.

Value

A list for each provided time point. See the help of mvslouchModel for what the summary at each time point is.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Pienaar, J. and Mostad. P. and Andersson, S. and Hansen, T. F. (2012) A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204-215.

Bartoszek, K. and Tredgett Clarke J. and Fuentes-Gonzalez, J. and Mitov, V. and Pienaar, J. and Piwczynski, M. and Puchalka, R. and Spalik, K. and Voje, K. L. (2024) Fast mvSLOUCH: Multivariate Ornstein-Uhlenbeck-based models of trait evolution on large phylogenies, Methods in Ecology and Evolution 15(9):1507-1515.

Butler, M.A. and A.A. King (2004) Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695.

Hansen, T.F. (1997) Stabilizing selection and the comparative analysis of adaptation. Evolution 51:1341-1351.

Hansen, T.F. and Bartoszek, K. (2012) Interpreting the evolutionary regression: the interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61(3):413-425.

Pienaar et al (in prep) An overview of comparative methods for testing adaptation to external environments.

See Also

hansen, ouchModel, simulOUCHProcPhylTree, parametric.bootstrap

Examples

```
RNGversion(min(as.character(getRversion()),"3.6.1"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
### We will first simulate a small phylogenetic tree using functions from ape.
### For simulating the tree one could also use alternative functions, e.g. sim.bd.taxa
### from the TreeSim package
phyltree<-ape::rtree(5)

## The line below is not necessary but advisable for speed
phyltree<-phyltree_paths(phyltree)

### Define a vector of regimes.
regimes<-c("small","small","large","small","small","large","large","large")

### Define the SDE parameters to be able to simulate data under the OUOU model.
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(9,0,0),c(0,5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
"large"=c(-1,1,0.5)),Syy=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[phyltree$tip.label,,drop=FALSE]

## Here we do not do any recovery step
OUOU.summary<-SummarizeOUCH(phyltree,OUOUdata,OUOUparameters,
regimes,t=c(1),dof=15)
RNGversion(as.character(getRversion()))

## Not run:  ##It takes too long to run this
## Now we take a less trivial simulation setup
### Recover the parameters of the OUOU model.
```

```
OUOUestim<-ouchModel(phyltree,OUOUdata,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive",maxiter=c(10,100))

### Summarize them.
OUOU.summary<-SummarizeOUCH(phyltree,OUOUdata,OUOUestim$FinalFound$ParamsInModel,
regimes,t=c(1),dof=OUOUestim$FinalFound$ParamSummary$dof)


### And finally bootstrap with particular interest in the evolutionary regression
OUOUbootstrap<-parametric.bootstrap(estimated.model=OUOUestim,phyltree=phyltree,
values.to.bootstrap=c("evolutionary.regression"),regimes=regimes,root.regime="small",
M.error=NULL,predictors=c(3),kY=NULL,numboot=5,Atype=NULL,Syytype=NULL,diagA=NULL)

## End(Not run)
```

Index