Package ‘mvSLOUCH’

May 16, 2018

Type Package
Title Multivariate Stochastic Linear Ornstein-Uhlenbeck Models for Phylogenetic Comparative Hypotheses
Version 1.3.4
Date 2018-05-16
Author Krzysztof Bartoszek <bartoszekkJ@gmail.com>
Maintainer Krzysztof Bartoszek <bartoszekkJ@gmail.com>
Description Fits multivariate Ornstein-Uhlenbeck types of models to continues trait data from species related by a common evolutionary history.
Depends R (>= 2.9.1), ouch, ape, numDeriv, mvtnorm, corpcor
License GPL (>= 2) | file LICENCE
LazyLoad yes
   matrixparametrizations.R modelparamstransform.R precalcs.R
   modelparamssmmary.R modelparams.R estimBM.R estimGLGC.R
   estimMAXLIK.R simulVasicekproc.R simulVasicekprocphyl.R
   ouch2slouch.mvsl.R make.states.mvsl.R evolmodelest.R getESS.R
Imports methods
NeedsCompilation no
Repository CRAN
Date/Publication 2018-05-16 10:58:27 UTC

R topics documented:

  mvSLOUCH-package ............................................. 2
  BrownianMotionModel ......................................... 5
  drawPhylProcess .............................................. 7
  estimate.evolutionary.model ................................. 9
  fitch.mvsl .................................................. 13
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

\[ dY(t) = -A(Y(t) - \Psi(t) - A^{-1}BX(t))dt + \Sigma_{yy}dB(t) \]
\[ dX(t) = \Sigma_{xx}dB(t) \]

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 bartoszekkj@gmail.com.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>mvSLOUCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>1.3.4</td>
</tr>
<tr>
<td>Date</td>
<td>2018-05-16</td>
</tr>
<tr>
<td>License</td>
<td>GPL (&gt;= 2)</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>
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

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

models that evolve on a phylogenetic tree.

The estimation functions are BrownianMotionModel, ouchModel (OUOU) and mvslouchModel (mvOUBM). The rely on a combination of least squares and numerical optimization techniques. A wrapper function for all of them is estimate.evansionary.model, in tries all three model 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 is required to be of the same type that the ouch package requires.

The package uses the functions .sym.par and .sym.unpar from the ouch package to parametrize symmetric matrices. The package also uses the internal structure of ouch’s (as of version 2.8.2) tree object.

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

Author(s)
Krzysztof Bartoszek Maintainer: <bartoszekkj@gmail.com>

References


Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions,
### eg. sim.bd.taxa from the TreeSim package
phyltree<-ape2ouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@modellabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

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

### Define SDE parameters to be able to simulate data under the different models.
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)))
OUOUparameters<-list(v0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(0.5,0,c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
"large"=c(-1,1,0.5)),Syx=rbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
OUMBparameters<-list(v0=matrix(c(1,-1),ncol=1,nrow=2),
B=matrix(c(2,-2),ncol=1,nrow=2),mPsi=cbind("small"=c(1,-1),"large"=c(-1,1)),
Syx=matrix(0,ncol=1,nrow=2),vx0=matrix(0,1,1),Sxx=matrix(1,1,1),
Syx=matrix(0,ncol=2,nrow=1))

### Now simulate the data and remove the values corresponding to the internal nodes.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx)
OUOUpdata<-simulOUCHProcPhylTree(phyltree,OUOUpparameters,regimes,NULL)
OUMBdata<-simulMVSLOUCHProcPhylTree(phyltree,OUMBparameters,regimes,NULL)
OUMBdata<-OUMBdata[-c:(phyltree@nnodes-phyltree@nterm),]

### Recover the parameters of the SDEs.
BMestim<-BrownianMotionModel(phyltree,BMdata)
OUOUestim<-ouchModel(phyltree,OUOUpdata,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive")
OUMBestim<-mvSLouchModel(phyltree,OUMBdata,2,regimes,Atype="DecomposablePositive",
Syytype="UpperTri",diagA="Positive")

### Use the wrapper function
estimResultsBM<-estimate.evolutionary.model(demotreeouch,BMdata,regimes=NULL,
root.regime=NULL,M.error=NULL,repets=3,model.setups=NULL,predictors=c(3),KY=2,
doPrint=TRUE)
estimResultsOUOU<-estimate.evolutionary.model(demotreeouch,OUOUpdata,regimes,root.regime="small",M.error=NULL,repets=3,model.setups=NULL,predictors=c(3),KY=2,
doPrint=TRUE)
estimResultsOUMB<-estimate.evolutionary.model(demotreeouch,OUMBdata,regimes,root.regime="small",M.error=NULL,repets=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
```
The `brownianmotionmodel` function uses maximum likelihood to fit parameters of a Brownian motion model evolving on the phylogeny. If there are no missing values or measurement error then it just calls the `brown()` function from the `ouch` package.

Usage

```
brownianmotionmodel(phyltree, data, predictors = NULL, M.error = NULL, calcCI = FALSE)
```

Arguments

- **phyltree**: The phylogeny in `ouch` format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a `nexus` file by the `read.nexus` function from the `ape` package and converted into the `ouch` format by `ouch`'s `ape2ouch` function. See the example of how to correct the internal nodes.

- **data**: A data frame with the rows corresponding to the species while the columns correspond to the traits. The rows can be named by species, if not then the order of the species has to be the same as the order in which the species are on the phylogeny.

- **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.

- **M.error**: An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:
  - a single number that will be on the diagonal of the covariance matrix
  - a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
  - a nxm element vector a diagonal matrix with this vector on the diagonal,
BrownianMotionModel

- a \( m \times m \) (number of variables) \times (number of variables) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,

- a list of length \( m \) (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations has same variance), vector (of length \( n \) for each observation), or full matrix,

- matrix of size \( mn \times mn \) (\( m \) - number of variables, \( n \) - number of observations) the measurement error covariance provided as is,

- NULL no measurement error

calcCI  An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

Details

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. If there is no measurement error it uses the brown function from the ouch package, otherwise it does a numerical optimization.

Value

- **ParamsInModel** A list with estimated model parameters. The elements are vX0 : the ancestral trate, and Sxx where t*Sxx

- **ParamSummary** A list with summary statistics with elements, StS the infinitesimal covariance matrix Sxx LogLik the log–likelihood, dof the degrees of freedom, m2loglik is \(-2\)log–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


See Also

`summarizeBM`, `simulBMProcPhylTree`

Examples

```r
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
## Not run: ##It takes too long to run this
phyltree<-ape::ouch(rtree(5))

### Correct the names of the internal node labels
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

### 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=rbinding(c(1,0,0),c(0.2,1,0),c(0.3,0.25,1)))

### Now simulate the data and remove the values corresponding to the internal nodes.
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$VX0,Sigma=BMparameters$Sxx)
BMdata<-BMdata[-(1:(phyltree@nnodes-phyltree@nterm)),]

### And finally try to recover the parameters of the Brownian motion.
BMestim<-BrownianMotionModel(phyltree,BMdata)

## End(Not run)
```

drawPhylProcess  

Plots the realization of a process evolving on a phylogenetic tree

drawPhylProcess(PhylTraitProcess, phyltree = 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.

**phyltree**
The phylogenetic tree, currently not used in the function but parameter is for future development

**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.

**xlims**
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**
Examples

```r
## Not run:  ##It takes too long to run this
## We will first simulate a small phylogenetic tree using functions from ape and ouch.
## For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
## from the TreeSim package
phyltree<-ape2ouch(rtree(5))

## Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

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

## Define SDE parameters to be able to simulate data under the ouou model.
OUOUparameters<-list(VV=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(0,0,0),c(0,0,0)),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 and remove the values corresponding to the internal nodes.
OUOUdata<-simuOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL,fullTrajectory=TRUE)
drawPhylProcess(PhylTraitProcess=OUOUdata)
```

esteem.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 and returns the best overall model.

Usage

```r
estimate.evolutionary.model(phyltree, dfdata, regimes = NULL,
root.regime = NULL, M.error = NULL, repeats = 3, model.setups = NULL,
predictors = NULL, kY = NULL, doPrint = FALSE, pESS=NULL,estimate.root.state=FALSE)
```

Arguments

- `phyltree` The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the ape package and converted into the ouch format by ouch’s `ape2ouch` function. See the example of how to correct the internal nodes. It is
strongly advisable to rescale the tree so that the determinant of the speciation
time matrix (phy1tree@branch.times) is about 1. This significantly stabilizes
the estimation procedure.

**dfdata**
A data frame with the rows corresponding to the species while the columns
correspond to the traits. The rows can be named by species, if not then the order
of the species has to be the same as the order in which the species are on the
phylogeny.

**regimes**
A vector or list of regimes. If vector then each entry corresponds to the branch
preceding the respective node. If list then each list entry corresponds to a 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 taken as first element of regimes.

**M.error**
An optional measurement error covariance matrix. The program tries to recog-
nizes the structure of matrix passed and accepts the following possibilities :

- a single number that will be on the diagonal of the covariance matrix
- a m element vector with each value corresponding to a variable and the
covariance matrix will have that vector repeated on its diagonal,
- a nxm element vector a diagonal matrix with this vector on the diagonal,
- a m x m ((number of variables) x (number of variables)) matrix it is assumed
  that the measurement errors are independent between observations so the
  resulting covariance structure is block diagonal,
- a list of length m (number of variables), each list element is the covariance
  structure for the appropriate variable, either a single number (each observa-
tions has same variance), vector (of length n for each observation), or full
  matrix,
- matrix of size mn x mn (m - number of variables, n - number of observa-
tions) the measurement error covariance provided as is,
- NULL no measurement error

**repeats**
How many starting points for the numerical maximum likelihood procedure
should be tried for each model setup.

**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 use is also free to provide their own list of models in this vari-
able. 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), "mvsloch" (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".

• `syttype` 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 ("Positive"), negative ("Negative") or the sign free to vary (NULL)

A minimum example list is `list(list(evolmodel="BM"))`.

**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. 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.

**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), "mvregr" (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 is "reg".

**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**
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.
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 standard errors. 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 slouch package is a recommended alternative if one has a OUBM models and only a single response (Y) trait. The ouch package considers an univariate OU model and looking at it could be helpful.

Author(s)

Krzysztof Bartoszek

References


See Also

`BrownianMotionModel, SummarizeBM, simulBMProcPhylTree hansen, ouchModel, SummarizeOUCH, simulOUCHProcPhylTree, slouch::model.fit, mvslouchModel, SummarizeMVSLOUCH, simulMVSLOUCHProcPhylTree`
Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
phyltree<-ape2ouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

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

### Define SDE parameters to be able to simulate data under the OU model.
OUparameters<-list(vY=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=rbind(c(0,0,0),c(0,5,0),c(0,0,1)),
M Psi=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 and remove the values corresponding to the internal nodes.
OUOdata<-simoulOCHProcPhylTree(phyltree,OUOparameters,regimes,NULL)
OUOdata<-OUOdata[-c(1:(phyltree@nnodes-phyltree@nterm))]

### And finally try to recover the parameters of the OU model.
estimResults<-estimate.evoluti.onal.model(demotreeouch,OUOdata,regimes=regimes,
 root.regime="small",M.error=NULL,repeats=3,model.setups=NULL,predictors=c(3),kY=2,
doPrint=TRUE,pESS=FALSE)
```

## 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

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

Arguments

- **phyltree** The phylogenetic tree in `ouch` format.
- **niche** The specific niche variable in the `slouch` data.frame to be reconstructed, entered as data.frame$niche
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 vector of reconstructed character states.

Author(s)

Jason Pienaar <jasonpienaar@gmail.com>

References


See Also

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

Examples

```r
phyltree<-ape2ouch(rtree(5))
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-
 as.character(1:(phyltree@nnodes-phyltree@nterm))
regimes<-c("A","B","B","C","C")
regimesFitch<-fitch.mvsl(phyltree,regimes,root=1,deltran=TRUE)
```
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(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.

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), "mvsloch" (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".
- 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


See Also

estimate.evolutionary.model
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.

Usage

mvslouchModel(phyltree, data, kY, regimes = NULL, regimes.times = NULL, root.regime = NULL, predictors = NULL, M.error = NULL, Atype = "Invertible", Syytype = "UpperTri", calcCI = FALSE, diagA = "Positive", estimate.root.state=FALSE)

Arguments

phyltree  The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the read.nexus function from the ape package and converted into the ouch format by ouch’s ape2ouch function. See the example of how to correct the internal nodes.

data  A data frame with the rows corresponding to the species while the columns correspond to the traits. The rows can be named by species, if not then the order of the species has to be the same as the order in which the species are on the phylogeny.

kY  Number of "Y" (response) variables.

regimes  A vector or list of regimes. If vector then each entry corresponds to the branch preceding the respective node. If list then each list entry corresponds to a 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 taken as first element of regimes.

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 the "X" variables are treated as predictors.

M.error  An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:

- a single number that will be on the diagonal of the covariance matrix
- a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
mvslouchModel

- A nxm element vector: a diagonal matrix with this vector on the diagonal,
- A m x m ((number of variables) x (number of variables)) matrix: it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
- A list of length m (number of variables), each list element: is the covariance structure for the appropriate variable, either a single number (each observation has same variance), vector (of length n for each observation), or full matrix,
- A matrix of size mn x mn (m - number of variables, n - number of observations): the measurement error covariance provided as is,
- NULL: no measurement error

**Atype**

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

**calcCI**
An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

**diagA**
Whether the values on A's diagonal are to be "Positive", "Negative" or sign allowed to vary, NULL.

**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**

This function estimates the parameters of the following multivariate SDE,

\[
\begin{align*}
\frac{dY(t)}{dt} &= -A(Y(t) - (\Psi(t) - A^{-1}BX(t)))dt + \Sigma_{yy}dB(t) \quad Y(0) = Y_0, \\
\frac{dX(t)}{dt} &= \Sigma_{xx}dB(t) \quad X(0) = X_0
\end{align*}
\]

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)B X(0) plus the root value of Psi.

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 value of mPsi0 can be currently ignored. The field ParamSummary are different composite (evaluated at time 1) and summary statistics. The field phylhalflife are the eigenvalues, eigenvectors and phylogenetic half lives associated with the A matrix of, expmtA is \(\exp(-A \ast 1)\), optimal regression is the -solve(A)B matrix, mPsi.rotated is each of the regime effects multiplied by
mvslouchModel

\[ 1 - \exp(-A \ast 1), \] mPsi0.rotated is the same for the mPsi0 vector and can be ignored, cov.matrix is the trait vector covariance matrix at time 1, corr.matrix is the trait vector correlation matrix at time 1, conditional.cov.matrix is the conditional covariance matrix of the OU type variables on the Brownian motion type at time 1, i.e. Cov[Y|X](1), conditional.corr.matrix is the conditional correlation matrix of the OU type variables on the Brownian motion type at time 1, i.e. Corr[Y|X](1), 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 time 1 equalling \(1\ast \text{solve}(A)S_{xx}(S_{xx}^T)(B^T)\text{solve}(A)^T\), optima.corr.matrix is the correlation matrix of the optimal process at time 1, 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](1) StS 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 last field LogLik is the log–likelihood at the point.

**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**


See Also

slouch::modelNfit, SummarizeMVSLOUCH, simulMVSLOUCHProcPhylTree

Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
phyltree<apeouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[:(phyltree@nnodes-phyltree@nterm)]<as.character(1:(phyltree@nnodes-phyltree@nterm))

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

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<:list(v0=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=rbind("small"=c(1,-1),"large"=c(-1,1)),
Sy0=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 and remove the values corresponding to the internal nodes.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[-(1:(phyltree@nnodes-phyltree@nterm))]

### And finally try to recover the parameters of the mvOUBM model.
OUBMrestim<mvslouchModel(phyltree,OUBMdata,2,regimes,Atype="DecomposablePositive",Sytype="UpperTri",diagA="Positive")

## End(Not run)
```
**Description**

The `ouchModel` function uses maximum likelihood to fit parameters of a multivariate OU model evolving on the phylogeny.

**Usage**

```r
ouchModel(phyltree, data, regimes = NULL, regimes.times = NULL,
root.regime = NULL, predictors = NULL, M.error = NULL, Atype = "Invertible",
Syytype = "UpperTri", calcCI = FALSE, diagA = "Positive", estimate.root.state=FALSE)
```

**Arguments**

- `phyltree` The phylogeny in `ouch` format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the `ape` package and converted into the `ouch` format by `ouch`'s `ape2ouch` function. See the example of how to correct the internal nodes.
- `data` A data frame with the rows corresponding to the species while the columns correspond to the traits. The rows can be named by species, if not then the order of the species has to be the same as the order in which the species are on the phylogeny.
- `regimes` A vector or list of regimes. If vector then each entry corresponds to the branch preceding the respective node. If list then each list entry corresponds to a 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 taken as first element of regimes.
- `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 the "X" variables are treated as predictors.
- `M.error` An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:
  - a single number that will be on the diagonal of the covariance matrix
  - a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
  - a nxm element vector a diagonal matrix with this vector on the diagonal,
  - a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
  - a m x m ((number of variables) x (number of variables)) block-diagonal matrix. The program tries to recognize the block-diagonal structure and use an appropriate algorithm to solve the likelihood equations.

...
• a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observation has same variance), vector (of length n for each observation), or full matrix,
• matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
• NULL no measurement error

**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"

**Syytype**

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

**calcCI**

An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

**diagA**

Whether the values on A's diagonal are to be "Positive", "Negative" or sign allowed to vary, NULL.

**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

This function estimates the parameters of the following multivariate SDE,

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

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.

The found point is described by a list containing four fields. The first field HeuristicSearchPoint-FinalFind 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 value of mPsi0 can be currently ignored. The field ParamSummary are different composite (evaluated at time 1) and summary statistics, The field phylhalflife are the eigenvalues, eigenvectors and phylogenetic half lives associated with the A matrix, expmtA is exp(-A*1), mPsi.rotated is each of the regime effects multiplied by (1-exp(-A*1)), mPsi0.rotated is the same for the mPsi0 vector and can be ignored, cov.matrix is the trait vector covariance matrix at time 1, corr.matrix is the trait vector correlation matrix at time 1, 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 S LogLik the log–likelihood, dof the degrees of freedom, m2loglik is −2log–likelihood, aic is the Akiake information criterion, aic.c is the Akiake 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 last field LogLik is the log–likelihood at the point.

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


See Also

hansen, SummarizeOUCH, simulOUCHProcPhylTree

Examples

## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
### 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

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

#### Arguments

- **phyltree**: The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the ape package and converted into the ouch format by ouch’s `ape2ouch` function. See the example of how to correct the internal nodes.

- **X0**: The ancestral, root state.

- **Sigma**: The diffusion matrix of the Brownian motion.

- **dropInternal**: Logical whether the simulated values at the internal nodes be changed to NA or not.
M.error  An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:

- a single number that will be on the diagonal of the covariance matrix
- a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
- a nxm element vector a diagonal matrix with this vector on the diagonal,
- a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
- a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations have same variance), vector (of length n for each observation), or full matrix,
- matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
- NULL no measurement error

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 desceding 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] indictating the probability of a jump taking place, only matters if jumotype 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.

Value

If fullTrajectory is FALSE then returns a data.frame 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.

Author(s)

Krzysztof Bartoszek
References


See Also

BrownianMotionModel, SummarizeBM

Examples

```r
## Not run:  ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
phyltree<-ape2ouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

### 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 and remove the values corresponding to the internal nodes.
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpprob="Normal",
vMean=rep(0,3),mCov=diag(1,3,3))
BMdata<-simulBMProcPhylTree(phyltree,X0=BMparameters$vX0,Sigma=BMparameters$Sxx,jumpsetup=jumpobj)

## End(Not run)```
Simulate data on a phylogeny under a (multivariate) OUBM model

### Description

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

### Usage

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

### Arguments

- **phyltree**: The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the ape package and converted into the ouch format by ouch’s `ape2ouch` function. See the example of how to correct the internal nodes.

- **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 the branch preceding the respective node. If list then each list entry corresponds to a 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 be changed to NA or not.

- **M.error**: An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:
  - a single number that will be on the diagonal of the covariance matrix
  - a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
  - a nxm element vector a diagonal matrix with this vector on the diagonal,
  - a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
  - a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations has same variance), vector (of length n for each observation), or full matrix,
• matrix of size $mn \times mn$ ($m$ - number of variables, $n$ - number of observations) the measurement error covariance provided as is,
• NULL no measurement error

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 "Both" 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.

Value

If fullTrajectory is FALSE then returns a data.frame 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.

Author(s)

Krzysztof Bartoszek

References


See Also

mvslouchmodel, SummarizeMVS LOUCH

Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch. 
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxon
### from the TreeSim package
phyltree<-ape2ouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

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

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-list(vXo=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)),
Syx=rbind(c(1,0.25),c(0,1)),vX0=matrix(0,1,1),Sxx=matrix(1,1,1),
Sxy=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data and remove the values corresponding to the internal nodes.
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpdistribs="Normal",
vMean=rep(0,3),mCov=diag(1,3))
OUBMdata<-simulMVS LOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL,jumpsetup=jumpobj)

## End(Not run)
```

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

**Description**

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

Usage

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

Arguments

**phyltree**  
The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the ape package and converted into the ouch format by ouch’s `apeRouch` function. See the example of how to correct the internal nodes.

**modelParams**  
List of model parameters of OUOU model as `ParamsInModel` part of output of `ouch`.

**regimes**  
A vector or list of regimes. If vector then each entry corresponds to the branch preceding the respective node. If list then each list entry corresponds to a 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 be changed to NA or not.

**M.error**  
An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:

- a single number that will be on the diagonal of the covariance matrix
- a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
- a nxm element vector a diagonal matrix with this vector on the diagonal,
- a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
- a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations has same variance), vector (of length n for each observation), or full matrix,
- matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
- NULL no measurement error

**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:

- jumptypeIn 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 "JumpWithProb2".
• 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.

Value

If fullTrajectory is FALSE then returns a data.frame 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.

Author(s)

Krzysztof Bartoszek

References


See Also

hansen, ouchModel, simulOUCHProcPhylTree

Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
```
### from the TreeSim package

`phyltree<-ape2ouch(rtree(5))`

### Correct the names of the internal node labels.

`phyltree$nodeLabels[1:(phyltree$nNodes-phyltree$nterm)]<-'as.character(1:(phyltree$nNodes-phyltree$nterm))`

### Define a vector of regimes.

`regimes<-c("small","small","small","large","small","large","large","large","large")`

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

```r
OUOUparameters<-list(vY0=matrix(c(1,-1,0.5),nrow=3,ncol=1),
A=cbind(c(9,0,0),c(0,5,0),c(0,0,1)),nPsi=cbind("small"=c(1,-1,0.5),"large"=c(-1,1,0.5)),Syy=cbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))
```

### Now simulate the data and remove the values corresponding to the internal nodes.

```r
jumpobj<-list(jumptype="RandomLineage",jumpprob=0.5,jumpdistrib="Normal", vMean=rep(0,3),mCov=diag(1,3,3))
OUOUdata<-simOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL,jumpsetup=jumpobj)
```

### End(Not run)
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.

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 matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:

• a single number that will be on the diagonal of the covariance matrix
• a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
• a nxm element vector a diagonal matrix with this vector on the diagonal,
• a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
• a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations has same variance), vector (of length n for each observation), or full matrix,
• matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
• NULL no measurement error

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.

calcCI

An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

Value

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

Warning

Calculating the confidence intervals can take a very long time. Warnings and errors (even of lot of them) can be produced during the confidence interval calculation, this is nothing to worry about.

Author(s)

Krzysztof Bartoszek

References

See Also

BrownianMotionModel, simulBMProcPhylTree

Examples

```r
## Not run:  ##It takes too long to run this
## We will first simulate a small phylogenetic tree using functions from ape and ouch.
## For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
## from the TreeSim package
phyltree<-ape2ouch(rtree(5))

## Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(1:(phyltree@nnodes-phyltree@nterm))

## 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 and remove the values corresponding to the internal nodes.
Bmdata<-simulBMProcPhylTree(phyltree,X0=Bmparameters$vx0,Sigma=Bmparameters$sxx)
Bmdata<-Bmdata[-(1:(phyltree@nnodes-phyltree@nterm)),]

## Recover the parameters of the Brownian motion.
Bmestim<-BrownianMotionModel(phyltree,Bmdata)

## And summarize them.
Bm.summary<-SummarizeBM(phyltree,Bmdata,Bmestim$ParamsInModel,t=c(1),
doF=Bmestim$ParamSummary$doF,calcCI=FALSE)

## if one would want the confidence intervals then set calcCI=TRUE

## End(Not run)
```
Summary of MVSLOUCH

Description

Compiles a summary (appropriate moments, conditional moments, information criteria) of parameters of a multivariate OUBM model at a given time point.

Usage

`SummarizeMVSLOUCH(phyltree, data, modelParams, regimes = NULL, regimes.times = NULL, t = c(1), dof = NULL, M.error = NULL, predictors = NULL, Atype = "Invertible", Syytype = "UpperTri", calcCI = FALSE)`

Arguments

- `phyltree`: The phylogeny in `ouch` format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the `ape` package and converted into the `ouch` format by `ouch`'s `ape2ouch` function. See the example of how to correct the internal nodes.
- `data`: A data frame with the rows corresponding to the species while the columns correspond to the traits. The rows can be named by species, if not then the order of the species has to be the same as the order in which the species are on the phylogeny.
- `modelParams`: A list of model parameters, as returned in `ParamsInModel` part of `mvsloouchmodel`'s output.
- `regimes`: A vector or list of regimes. If vector then each entry corresponds to the branch preceding the respective node. If list then each list entry corresponds to a 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.
- `dof`: Number of unknown parameters in the model, can be extracted from the output of `mvsloouchModel()`. If not provided all parameters are assumed unknown.
- `M.error`: An optional measurement error covariance matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:
  - a single number that will be on the diagonal of the covariance matrix
  - a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
  - a nxm element vector a diagonal matrix with this vector on the diagonal,
  - a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
• a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observation has same variance), vector (of length n for each observation), or full matrix,
• matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
• NULL no measurement error

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.

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"

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

calcCI
An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

Details

If calcCI is set to TRUE the function returns the confidence intervals, the log-likelihood surface for the eigenvalues conditional on the other parameters and regression confidence intervals. See Bartoszek et. al. for the mathematical details of these confidence intervals.

Value

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

Warning

Calculating the confidence intervals can take a very long time. Warnings and errors (even of lot of them) can be produced during the confidence interval calculation, this is nothing to worry about.

Author(s)

Krzysztof Bartoszek

References


See Also

slouch::modelNfit, mvslouchModel, simulMVSLOUCHProcPhylTree

Examples

```r
## Not run: ##It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.taxa
### from the TreeSim package
phyltree<-ape2ouch( rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character( 1:(phyltree@nnodes-phyltree@nterm))

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

### Define SDE parameters to be able to simulate data under the mvOUBM model.
OUBMparameters<-(list(vY=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=rbind("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),
Sxy=matrix(0,ncol=1,nrow=2),Sxy=matrix(0,ncol=2,nrow=1))

### Now simulate the data and remove the values corresponding to the internal nodes.
OUBMdata<-simulMVSLOUCHProcPhylTree(phyltree,OUBMparameters,regimes,NULL)
OUBMdata<-OUBMdata[-(-1:(phyltree@nnodes-phyltree@nterm)),]

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

### And summarize them.
OUBM_summary<-SummarizeMVSLOUCH(phyltree,OUBMdata,OUBMestim$FinalFound$ParamsInModel, regimes,t=1,dof=OUBMestim$FinalFound$ParamSummary$dof,calcCI=FALSE)
### if one would want the confidence intervals then set calcCI=TRUE

## End(Not run)
```
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.

### Usage

```r
SummarizeOUCH(phyltree, data, modelParams, regimes = NULL, regimes.times = NULL, t = c(1), dof = NULL, M.error = NULL, predictors = NULL, Atype = "Invertible", Syytype = "UpperTri", calcCI = FALSE)
```

### Arguments

- **phyltree**
  The phylogeny in ouch format. All of the internal nodes have to be uniquely named. The tree can be obtained from e.g. a nexus file by the `read.nexus` function from the ape package and converted into the ouch format by ouch’s `ape2ouch` function. See the example of how to correct the internal nodes.

- **data**
  A data frame with the rows corresponding to the species while the columns correspond to the traits. The rows can be named by species, if not then the order of the species has to be the same as the order in which the species are on the phylogeny.

- **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 the branch preceding the respective node. If list then each list entry corresponds to a 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.

- **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 matrix. The program tries to recognizes the structure of matrix passed and accepts the following possibilities:
  - a single number that will be on the diagonal of the covariance matrix
  - a m element vector with each value corresponding to a variable and the covariance matrix will have that vector repeated on its diagonal,
• a nxm element vector a diagonal matrix with this vector on the diagonal,
• a m x m ((number of variables) x (number of variables)) matrix it is assumed that the measurement errors are independent between observations so the resulting covariance structure is block diagonal,
• a list of length m (number of variables), each list element is the covariance structure for the appropriate variable, either a single number (each observations has same variance), vector (of length n for each observation), or full matrix,
• matrix of size mn x mn (m - number of variables, n - number of observations) the measurement error covariance provided as is,
• NULL no measurement error

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.

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"

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

calcCI An optional logical variable indicating whether to calculate and return confidence intervals on the estimated parameters.

Details

If calcCI is set to TRUE the function returns the confidence intervals, the log-likelihood surface for the eigenvalues conditional on the other parameters and regression confidence intervals. See Bartoszek et. al. for the mathematical details of these confidence intervals.

Value

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

Warning

Calculating the confidence intervals can take a very long time. Warnings and errors (even of lot of them) can be produced during the confidence interval calculation, this is nothing to worry about.

Author(s)

Krzysztof Bartoszek

References


See Also

hansen, ouchModel, simulOUCHProcPhylTree

Examples

```r
# Not run: #It takes too long to run this
### We will first simulate a small phylogenetic tree using functions from ape and ouch.
### For simulating the tree one could also use alternative functions, eg. sim.bd.t taxa
### from the TreeSim package
phyltree<-ape2ouch(rtree(5))

### Correct the names of the internal node labels.
phyltree@nodelabels[1:(phyltree@nnodes-phyltree@nterm)]<-as.character(
  1:(phyltree@nnodes-phyltree@nterm))

### Define a vector of regimes.
regimes<-c("small","small","small","large","small","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=cbind(c(0,0,0),c(0.5,0),c(0,0,1)),mPsi=cbind("small"=c(1,-1,0.5),
  "large"=c(-1,1,0.5)),Syy=cbind(c(1,0.25,0.3),c(0,1,0.2),c(0,0,1)))

### Now simulate the data and remove the values corresponding to the internal nodes.
OUOUdata<-simulOUCHProcPhylTree(phyltree,OUOUparameters,regimes,NULL)
OUOUdata<-OUOUdata[-1:(phyltree@nnodes-phyltree@nterm),]

### Recover the parameters of the OUOU model.
OUOUestim<-ouchModel(phyltree,OUOUdata,regimes,Atype="DecomposablePositive",
  Syytype="UpperTri",diagA="Positive")

### And summarize them.
OUOU.summary<-SummarizeOUCH(phyltree,OUOUdata,OUOUestim$FinalFound$ParamsInModel,
  regimes,t=c(1),dof=OUOUestim$FinalFound$ParamSummary$dof,calcCI=FALSE)
### if one would want the confidence intervals then set calcCI=TRUE

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

*Topic **hplot**
drawPhylProcess, 7

*Topic **models**
BrownianMotionModel, 5
drawPhylProcess, 7
estimate.evolutionary.model, 9
fitch.mvsl, 13
generate.model.setups, 15
mvSL OUCH-package, 2
mvslouchModel, 16
ouchModel, 20
simulBMProcPhylTree, 23
simulMVSL OUCHProcPhylTree, 26
simulOUCHProcPhylTree, 28
SummarizeBM, 31
SummarizeMVSL OUCH, 33
SummarizeOUCH, 37

BrownianMotionModel, 5, 12, 25, 32, 33
drawPhylProcess, 7
estimate.evolutionary.model, 9, 15
fitch.mvsl, 13
generate.model.setups, 15
hansen, 12, 22, 30, 39
mvSL OUCH (mvSL OUCH-package), 2
mvSL OUCH-package, 2
mvslouchModel, 12, 16, 28, 35, 36, 38
ouchModel, 12, 20, 30, 39
simulBMProcPhylTree, 7, 12, 23, 33
simulMVSL OUCHProcPhylTree, 12, 19, 26, 36
simulOUCHProcPhylTree, 12, 22, 28, 30, 39
SummarizeBM, 7, 12, 25, 31
SummarizeMVSL OUCH, 12, 19, 28, 33
SummarizeOUCH, 12, 22, 37