Package 'pcmabc'

May 9, 2026

Type Package

Title Approximate Bayesian Computations for Phylogenetic Comparative Methods

Version 1.1.3

Date 2022-05-06

Author Krzysztof Bartoszek <krzbar@protonmail.ch>, Pietro Lio'

Maintainer Krzysztof Bartoszek <krzbar@protonmail.ch>

Description Fits by ABC, the parameters of a stochastic process modelling the phylogeny and evolution of a suite of traits following the tree. The user may define an arbitrary Markov process for the trait and phylogeny. Importantly, trait-dependent speciation models are handled and fitted to data. See K. Bartoszek, P. Lio' (2019) <doi:10.5506/APhysPolBSupp.12.25>. The suggested geiger package can be obtained from CRAN's archive <https://cran.r-project.org/src/contrib/Archive/geiger/>, suggested to take latest version. Otherwise its required code is present in the pcmabc package. The suggested distory package can be obtained from CRAN's archive <https://cran.r-project.org/src/contrib/Archive/distory/>, suggested to take latest version.

Depends R(>= 2.9.1)

Imports ape (>= 3.0.6), graphics, methods, mvSLOUCH, phangorn, stats, utils, yuima

Suggests geiger, distory

License GPL (>= 2) | file LICENCE

LazyLoad yes

Collate ABCdist.R ABCmessage.R ABC.R ABCupdate.R draw.R from_geiger_2_0_6_2_diversification.R getTipSample.R rates.R simsde.R simtree.R

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-09 08:40:07 UTC

Contents

`pcmabc-package` *Approximate Bayesian Computations for Phylogenetic Comparative Methods*

Description

The package allows for Approximate Bayesian Computations (ABC) inference and simulation of stochastic processes evolving on top of a phylogenetic tree. The user is allowed to define their own stochastic process for the trait(s), be they univariate, multivariate continuous or discrete. The traits are allowed to influence the speciation and extinction rates generating the phylogeny. The user provides their own function that calculates these rates and one of the functions parameters is the current state of the phenotype. Two functionalities that are missing at the moment is for the speciation and extinction rates to be time-inhomogenous and that speciation events can influence the phenotypic evolution. It is planned to add this functionality. However, cladogenetic dynamics are possible at the start of the lineage, i.e. when a new lineage is separated from the main lineage. Hence, cladogenetic change can only be included as an event connected with a lineage (subpopulation) breaking off.

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:	pcmabc
Type:	Package
Version:	1.1.3
Date:	2022-05-06
License:	GPL (>= 2)
LazyLoad:	yes

The package allows for Approximate Bayesian Computations (ABC) inference and simulation of stochastic processes evolving on top of a phylogenetic tree. The `PCM_ABC()` function is responsible

for the inference procedure. The user has to provide their own functions to simulate the phenotype and tree. The package provides simulation of the trait under stochastic differential equation (SDE) models by calling **yuima**. In this case the user has to provide their own wrapper that creates an object **yuima** will be able to handle (see Example). The user also has to provide a function to calculate the instantaneous birth and death rates (or state that the tree is independent of the trait). The package supports some simple birth rate functions, see description of `PCM_ABC()`.

One is allowed to simulate a trait process and tree process jointly (with the trait influencing the tree's dynamics). This is done by the function `simulate_phylproc()`. The function `simulate_phenotype_on_tree()` simulates a trait process on top of a provided phylogeny. Finally, the function `simulate_sde_on_branch()` allows one to simulate an SDE model along a time interval using **yuima**. The trajectory of the trait(s) can be visualized using `draw_phylproc()`.

Author(s)

Krzysztof Bartoszek, Pietro Lio' Maintainer: <krzbar@protonmail.ch>

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

Kutsukake N., Innan H. (2014) Detecting Phenotypic Selection by Approximate Bayesian Computation in Phylogenetic Comparative Methods. In: Garamszegi L. (eds) Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology. Springer, Berlin, Heidelberg

Sheldon R. M. (2006). Simulation. Elsevier Academic Press.

Examples

```
## simulate 3d OUBM model
## This example requires the R package ape
## (to generate the tree in phylo format).
set.seed(12345)

phyltree<-ape::rphylo(n=5,birth=1,death=0)

simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838)
X0<-c(5.723548,4.103157,3.834698)
```

```
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
          ## in reality should be much smaller e.g. step<-0.001

simres<-simulate_phenotype_on_tree(phyltree, simulate_mvsl_sde, sde.params, X0, step)

## visualize the simulation
draw_phylproc(simres)

## extract the measurements at the tips
phenotypedata<-get_phylogenetic_sample(simres)

birth.params<-list(scale=1,maxval=2,abcstepsd=0.1,positivevars=c(TRUE,TRUE),
fixed=c(FALSE,TRUE))

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838,
positivevars=c(TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,FALSE),
abcstepsd=rep(0.1,12))

par0<-list(phenotype.model.params=sde.params,birth.params=birth.params)
fbirth<-"rate_id" ## should be rate_const but used here as an example
## for birth.params
fdeath<-NULL
X0<-c(5.723548,4.103157,3.834698)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
          ## in reality should be much smaller e.g. step<-0.001
abcsteps<-2 ## for keeping example's running time short <5s as CRAN policy,
            ## in reality should be much larger e.g. abcsteps<-500
eps<-1 ## for toy example's output to be useful,
       ## in reality should be much smaller e.g. eps<-0.25

## estimate parameters
ABCres<-PCM_ABC(phyltree=phyltree,phenotypedata=phenotypedata,
par0=par0,phenotype.model=simulate_mvsl_sde,fbirth=fbirth,fdeath=fdeath,X0=X0,
step=step,abcsteps=abcsteps,eps=eps)
```

`draw_phylproc` *Plots the trajectory of the stochastic process that evolved on the phylogeny.*

Description

The function plots the trajectory of a stochastic process that evolved on top of a phylogeny.

Usage

```
draw_phylproc(simulobj)
```

Arguments

simulobj	The simulate data as generated by simulate_phyloproc or simulate_phenotype_on_tree.

Details

The function is essentially a wrapper around mvSLOUCH::drawPhylProc(). It transforms the simulobj into a matrix understandable by mvSLOUCH::drawPhylProc() and then calls mvSLOUCH::drawPhylProc().

Value

Returns a meaningless NA value.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

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

```
set.seed(12345)

simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}
birth.params<-list(scale=1,maxval=2)
sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838)
X0<-c(5.723548,4.103157,3.834698)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
  ## in reality should be much smaller e.g. step<-0.001
simres<-simulate_phylproc(3.5, sde.params, X0, fbirth="rate_id", fdeath=NULL,
fbirth.params=NULL, fdeath.params=NULL, fsimulphenotype=simulate_mvsl_sde,
n.contemporary=5, n.tips.total=-1, step=step)
```

```
draw_phylproc(simres)
```

`get_phylogenetic_sample` *Retrieve contemporary sample*

Description

The function retrieved the contemporary sample from an object simulated by **pcmabc**.

Usage

```
get_phylogenetic_sample(pcmabc_simulobj, bOnlyContemporary=FALSE,
tol=.Machine$double.eps^0.5)
```

Arguments

`pcmabc_simulobj`	The output simulated object by `simulate_phylproc()` or `simulate_phenotype_on_tree()`.
`bOnlyContemporary`	Logical, should all tip measurements be extracted (`FALSE`) or only those corresponding to non-extinct tips (`TRUE`). In the latter case be careful with the output (see Value).
`tol`	The tolerance to check if the tip is contemporary, i.e. the tip is distant from the root by tree.height +/- `tol`.

Value

The function returns a matrix with rows corresponding to tips. If there are extinct species, but `bOnlyContemporary` was set to `TRUE`, then the extinct lineages have to be removed from the tree before any further analysis. Also after removing extinct lineages one has to make sure that the order of the contemporary nodes did not change in the tree. Otherwise, the rows of the output matrix will not correspond to the indices of the tree's tips.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

Examples

```
## simulate 3d OUBM model
## This example requires the R package ape
## (to generate the tree in phylo format).
set.seed(12345)

phyltree<-ape::rphylo(n=5,birth=1,death=0)
simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}
phenotype.model<-simulate_mvsl_sde

birth.params<-list(scale=1,maxval=10000,abcstepsd=1,positivevars=c(TRUE),
fixed=c(FALSE,TRUE,TRUE,TRUE,TRUE))

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838)
X0<-c(5.723548,4.103157,3.834698)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
          ## in reality should be much smaller e.g. step<-0.001
simres<-simulate_phenotype_on_tree(phyltree, simulate_mvsl_sde, sde.params, X0, step)
mPhylSamp<-get_phylogenetic_sample(simres)
```

PCM_ABC *ABC estimation for PCMs*

Description

The function does parameter estimation through Approximate Bayesian Computations (ABC) for user defined models of trait and phylogeny evolution. In particular the phenotype may influence the branching dynamics.

Usage

```
PCM_ABC(phyltree, phenotypedata, par0, phenotype.model, fbirth, fdeath = NULL,
X0 = NULL, step = NULL, abcsteps = 200, eps = 0.25, fupdate = "standard",
typeprintprogress = "dist", tree.fixed=FALSE,
dist_method=c("variancemean","wRFnorm.dist"))
```

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.
phenotypedata	A matrix with the rows corresponding to the tip species while the columns correspond to the traits. The rows should be in 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).
par0	The starting parameters for the estimation procedure. This is a list of 1, 2 or 3 lists. The lists have to be named as phenotype.model.params, birth.params (optional if tree.fixed is TRUE and death.params (optional). The phenotype.model.params list corresponds to parameters governing the trait evolution process, the birth.params to the birth rate of the branching process and the death.params to the extinction rate. The last element is optional as one can have just a pure birth tree. The entries of all the lists should be named. Each of the three lists can have three special fields fixed, abcstepsd, positivevars. The field fixed is a logical vector of length equalling the number of parameters. If an entry is TRUE, then that parameter is not optimized over but kept at its initial value throughout the whole ABC procedure. The field abcstepsd is a numeric vector equalling the number of parameters. It is the standard deviation of the random update of the parameter, i.e. providing control on how much one wants to jump in each parameter dimension. The field positivevars is a logical vector of length equalling the number of parameters and indicating if the given parameter is to be positive TRUE or arbitrary FALSE. Notice that even if fixed has true entries, then corresponding entries have to be present in abcstepsd and positivevars (but their values do not matter).
phenotype.model	The name of a function to simulate the phenotype over a period of time. The function has to have four parameters (in the following order not by name): time, params, X0 and step. The parameter time is the duration of the simulation, i.e. the length of the branch. The parameter params is a list of the parameters governing the simulation, what will be passed here is the list phenotype.model.params, without the fields fixed, abstepsd and positivevars. It is up to the function in phenotype.model to interpret the provided parameters. X0 stands for the value at the start of the branch and step is a control parameter governing the time step size of the simulation. The **pcmabc** package has inbuilt support for simulating the trait as as stochastic differential equation by the **yuima** package with the function simulate_sde_on_branch(). However, the user needs to write the phenotype.model function that translates the vector of parameters into an object that is understandable by **yuima** and then call simulate_sde_on_branch(), see the Example. The phenotype is simulated prior to the simulation of the speciation/extinction events on a lineage (see Details). Hence, it is not possible (at the moment) to include some special event (e.g. a cladogenetic jump) at branching. Such dynamics are only possible at the start of the lineage, i.e. when the new lineage

is separated from the main lineage. Hence, cladogenetic change can only be included as an event connected with a lineage (subpopulation) breaking off.

`fbirth` A function that returns the birth rate at a given moment. The fist parameter of the function has to correspond to the value of the phenotype (it is a vector first element is time and the others the values of the trait(s)) and the second to the list of birth parameters `birth.params`, see `par0`. The time entry of the phenotype vector is at the moment relative to an unspecified (from `fbirth`'s perspective) speciation event on the phylogeny. Hence, it cannot be used as for writing a time-inhomogenous speciation function. The speciation process is assumed to be time homogeneous in the current implementation. The package has support for two inbuilt rate functions. The can be indicated by passing a string in `fbirth`: either `"rate_id"` or `"rate_const"`. The string `"rate_const"` corresponds to a constant rate and has to have the rate's value in field called `$rate`. However, a switching of rates is allowed. If the value of the first trait exceeds a certain threshold (provided in field `$switch` of birth parameters), then the rate is changed to the value in `$rate2`, see body of hidden function `.f_rate_const()`, in file `rates.R`. The string `"rate_id"` corresponds to the `.f_rate_id()` function, in file `rates.R`. If the birth parameters are `NULL`, then the rate equals the value of the first phenotype. However, a number of linear, threshold and power transformations of the rate are possible. The field `varnum` indicates the index of the variable to take as the one influencing the rate (remember to add 1 for the time entry). Then, if x stands for the trait influencing the branching rate it is transformed into a rate by the following fields in the following order. Set `rate<-x` and let `params` correspond to the list containing the branching parameters.

- `substract`baserate<-max(0,params$rate-substract`base)` `rate<-abs(rate)`
- `p` and if `is.null(params$raise.at.end) || !params$raise.at.end` rate<-rate^params$p
- `base` and `!is.null(params$const)`
  `if (res<params$base){rate<-params$const}`
- `base` and `is.null(params$const)`
  `if (res<params$base){rate<-0}`
- `invbase` and `!is.null(params$const)`
  `if (res>params$invbase){rate<-params$const}`
- `invbase` and `is.null(params$const)`
  `if (res>params$invbase){rate<-0}`
- `scale`rate<-rate/params$scale
- `p` and if `params$raise.at.end`rate<-rate^params$p `res<-abs(res)`
- `maxval`if(rate>params$maxval){rate<-params$maxval}

`fdeath` A function that returns the birth rate at a given moment. Its structure is the same as `fbirth`. The current version of the package does not provide any support for any inbuilt function.

`X0` The value of the ancestral state at the start of the tree. If `NULL`, then the mean of the contemporary observations is used.

`step` The time step size for simulating the phenotype. If not provided, then calculated as `min(0.001,tree.height/1000)`.

abcsteps The number of steps of the ABC search procedure.

eps The acceptance tolerance of the ABC algorithm.

fupdate How should the parameters be updated at each step of the ABC search algorithm. The user may provide their own function that has to handle the following call `fupdate(par,par0,baccept,ABCdist,phenotypedata,phyltree)`, where `par` is the current proposal for the parameter, `baccept` a logical variable indicating if `par` were accepted or rejected and `ABCdist` the distance between the observed and simulated under `par` data. The three other parameters `par0`, `phenotypedata` and `phyltree` are those that were provided in the call to `ABCdist`. The user may write `"standard"` in place of providing a function and then the internal function `.f_standard_update()`. This function makes mean 0 normal jitters (with standard deviation provided through `abcstepsd` from `par0`) for accepted parameters or if they were not accepted draws new parameters from a uniform on [-10,10] distribution.

typeprintprogress What should be printed out at each step of the ABC search algorithm. If `NULL`, then nothing, otherwise the package offers one possibility, `"dist"`-the iteration number and distance of the simulated dataset from. The user is free to write their own function here. The first parameter of the function has to be an integer(the iteration number), the second a real value (the distance) and the third a list (the proposed model parameters, see `par0` format).

tree.fixed Does the trait value process influence the branching dynamics (`FALSE`) or the branching structure is independent of the trait (`TRUE`).

dist_method A vector with two entries, the first is the method for calculating the distance between the simulated and observed trait data. The second is is the method for calculating the distance between the simulated and observed phylogeny. Possible values for the phenotype distance are `"variance"`, `"variancemean"`, `"order"` and for the distance between phylogenies are `"bdcoeffs"`, `"node_heights"`, `"logweighted_node_heights"`, `"RF.dist"`, `"wRF.dist"`, `"wRFnorm.dist"`, `"KF.dist"`, `"path.dist"`, `"path.dist.weights"`, `"dist.topo.KF1994"`, `"treeDist"`, `"BHV"` and `"BHVedge"`. The `"BHV"` and `"BHVedge"` methods will only work if the `distory` package is installed. If it is not, then they will be replaced by `"wRFnorm.dist"` and `"RF.dist"` respectively. The `distory` package is orphaned at the moment on CRAN. See Details for description of the methods. The choice of which distance calculation method is better seems to depend on the model of evolution, number of `abcsteps` and value of `eps`. Some experimentation is recommended. If the tree is assumed to be fixed, then the tree distance method is ignored.

Details

Some details of the function might change. In a future release it should be possible for the user to provide their own custom distance function, time-inhomogenous branching and trait simulation. The fields `sum.dists.from.data` and `sum.inv.dists.from.data` will probably be removed from the output object.

At the moment the distance function is calculated as (tree.distance+trait.distance)/2 or only with trait.distance if the tree is assumed fixed. The possible distance functions between simulates and observed phenotypes are

- "variance"calculates the Euclidean distance between covariance matrices estimated from simulated and original data. The differences between entries are weighted by the sum of their absolute values so that the distance is in [0,1],
- "variancemean"calculates the Euclidean distance between mean vectors and covariance matrices estimated from simulated and original data. The differences between entries are weighted by the sum of their absolute values so that the distance is in [0,1]. The difference between the means and covariances are weighted equally.
- "order"calculates the mean squared difference between ordered (by absolute value) tip measurements (scaled by maximum observed trait in each dimension so that distance is in [0,1] and then the difference between each pair of traits is scaled by the sum of their absolute values to remove effects of scale).

The possible distance functions between simulates and observed phenotypes are

- "bdcoeffs"first using geiger::bd.km() estimates the net diversification rate for both trees and then calculates the distance between the trees as the total variation distance between two exponential distributions with rates equalling the estimated net diversification rates.
- "node_heights"calculates the root mean square distance between node heights (scaled by tree height so that distance is in [0,1] and then the difference between each node height is scaled by the sum of the two heights to remove effects of scale)
- "logweighted_node_heights"similarly as "node_heights" but additionally divides the squared scaled difference is divided by the logarithm of the inverse order (i.e. the highest is 1) statistic, to reduce the role of smaller heights.
- "RF.dist"calls phangorn::RF.dist()
- "wRF.dist"calls phangorn::wRF.dist() with normalize=FALSE
- "wRFnorm.dist"calls phangorn::wRF.dist() with normalize=TRUE
- "KF.dist"calls phangorn::KF.dist()
- "path.dist"calls phangorn::path.dist() with use.weight=FALSE
- "path.dist.weights"calls phangorn::path.dist() with use.weight=TRUE
- "dist.topo.KF1994"calls ape::dist.topo() with method="score"
- "BHV"calls distory::dist.multiPhylo() with method="geodesic"
- "BHVedge"calls distory::dist.multiPhylo() with method="edgeset"

The tree is simulated by means of a Cox process (i.e. Poisson process with random rate). First the trait is simulated along the spine of a tree, i.e. a lineage of duration tree.height. Then, along this spine the birth and death rates are calculated (they may depend on the value of the phenotype). The maximum for each rate is calculated and a homogeneous Poisson process with the maximum rate is simulated. Then, these events are thinned. Each event is retained with probability equalling true rate divided by maximum of rate (p. 32, Sheldon 2006). All speciation events are retained until the first death event.

Value

param.estimate A list, in the form of par0, with a point estimate of the parameters. This point estimate is calculated from all the accepted points by using inverse distance weighting. The distances for the weighting are the

`all.accepted` A list with all the accepted parameters during the ABC search. This will allow for the presentation of the posterior distribution of the parameters.

`sum.dists.from.data`
The sum of all the distances between the observed data and the simulated data under the accepted parameter sets in the ABC search procedure.

`sum.inv.dists.from.data`
The sum of all the inverses of the distances between the observed data and the simulated data under the accepted parameter sets in the ABC search procedure.

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

Sheldon R. M. (2006). Simulation. Elsevier Academic Press.

See Also

`simulate_sde_on_branch`

Examples

```
## simulate 3d OUBM model
## This example requires the R package ape
## (to generate the tree in phylo format).
set.seed(12345)

phyltree<-ape::rphylo(n=15,birth=1,death=0)

simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=10,psi2=-10)
X0<-c(10,4.103157,3.834698)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
           ## in reality should be much smaller e.g. step<-0.001
simres<-simulate_phenotype_on_tree(phyltree, simulate_mvsl_sde, sde.params, X0, step)
```

```
## extract the measurements at the tips
phenotypedata<-get_phylogenetic_sample(simres)

birth.params<-list(rate=10,maxval=10,abcstepsd=0.01,positivevars=c(TRUE,TRUE),
fixed=c(FALSE,TRUE))

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=10,psi2=-10,
positivevars=c(TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,FALSE),
abcstepsd=rep(0.1,12))

par0<-list(phenotype.model.params=sde.params,birth.params=birth.params)
fbirth<-"rate_const"
fdeath<-NULL
X0<-c(10,4.103157,3.834698)
step<-0.05 ## for keeping example's running time short <5s as CRAN policy,
           ## in reality should be much smaller e.g. step<-0.001
abcsteps<-2 ## for keeping example's running time short <5s as CRAN policy,
         ## in reality should be much larger e.g. abcsteps<-500
eps<-1 ## for toy example's output to be useful,
         ## in reality should be much smaller e.g. eps<-0.25
## estimate parameters
ABCres<-PCM_ABC(phyltree=phyltree,phenotypedata=phenotypedata,
par0=par0,phenotype.model=simulate_mvsl_sde,fbirth=fbirth,fdeath=fdeath,X0=X0,
step=step,abcsteps=abcsteps,eps=eps)
```

`simulate_phenotype_on_tree` *Simulate a stochastic process on top of a phylogeny.*

Description

The function simulates phenotypic dataset under user defined models trait evolution. The phenotype evolves on top of a fixed phylogeny.

Usage

```
simulate_phenotype_on_tree(phyltree, fsimulphenotype, simul.params, X0, step)
```

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`.

`fsimulphenotype`
: The name of a function to simulate the phenotype over a period of time. The function has to have four parameters (in the following order not by name): `time`, `params`, `X0` and `step`. The parameter `time` is the duration of the simulation, i.e. the length of the branch. The parameter `params` is a list of the parameters governing the simulation, what will be passed here is the list `phenotype.model.params`, without the fields `fixed`, `abstepsd` and `positivevars`. It is up to the function in `phenotype.model` to interpret the provided parameters. `X0` stands for the value at the start of the branch and `step` is a control parameter governing the time step size of the simulation. The **pcmabc** package has inbuilt support for simulating the trait as as stochastic differential equation by the **yuima** package with the function `simulate_sde_on_branch()`. However the user needs to write the `phenotype.model` function that translates the vector of parameters into an object that is understandable by **yuima** and then call `simulate_sde_on_branch()`, see the Example.

  The phenotype is simulated prior to the simulation of the speciation/extinction events on a lineage. Hence, it is not possible (at the moment) to include some special event (e.g. a cladogenetic jump) at branching. Such dynamics are only possible at the start of the lineage, i.e. when the new lineage is separated from the main lineage. Hence, cladogenetic change can only be included as an event connected with a lineage (subpopulation) breaking off.

`simul.params`
: The parameters of the stochastic model to simulate the phenotype. They should be passed as a named list.

`X0`
: The value of the ancestral state at the start of the tree.

`step`
: The time step size for simulating the phenotype. If not provided, then calculated as `min(0.001,tree.height/1000)`.

Value

`tree`
: The simulated tree in `phylo` format.

`phenotype`
: A list of the trajectory of the simulated phenotype. Each entry corresponds to a lineage that started at some point of the tree. Each entry is a matrix with first row equalling time (relative to start of the lineage, hence if absolute time since tree's origin is needed it needs to be recalculated, see `draw_phylproc()`'s code) and the next rows correspond to the trait value(s).

`root.branch.phenotype`
: The simulation on the root branch. A matrix with first row being time and next rows the simulated trait(s).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

See Also

simulate_phylproc

Examples

```
## simulate 3d OUBM model
## This example requires the R package ape
## (to generate the tree in phylo format).
set.seed(12345)

phyltree<-ape::rphylo(n=5,birth=1,death=0)
simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}

birth.params<-list(scale=1,maxval=10000)

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838)
X0<-c(5.723548,4.103157,3.834698)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
          ## in reality should be much smaller e.g. step<-0.001
simres<-simulate_phenotype_on_tree(phyltree, simulate_mvsl_sde, sde.params, X0, step)
```

---

simulate_phylproc *Simultaneously simulate a phylogeny and a trait evolving on it.*

Description

The function does simulates a phylogeny and phenotypic dataset under user defined models of trait and phylogeny evolution. In particular the phenotype may influence the branching dynamics.

Usage

```
simulate_phylproc(tree.height, simul.params, X0, fbirth, fdeath=NULL,
fbirth.params=NULL, fdeath.params=NULL, fsimulphenotype="sde.yuima",
n.contemporary=-1, n.tips.total=1000, step=NULL)
```

Arguments

tree.height	The height of the desired output tree. The simulated tree is conditioned to be of a certain height.
simul.params	The parameters of the stochastic model to simulate the phenotype. They should be passed as a named list.
X0	The value of the ancestral state at the start of the tree.
fbirth	A function that returns the birth rate at a given moment. The fist parameter of the function has to correspond to the value of the phenotype (it is a vector first element is time and the others the values of the trait(s)) and the second to the list of birth parameters birth.params, see par0. The time entry of the phenotype vector is at the moment relative to an unspecified (from fbirth's perspective) speciation event on the phylogeny. Hence, it cannot be used as for writing a time-inhomogenous speciation function. The speciation process is assumed to be time homogeneous in the current implementation. The package has support for two inbuilt rate functions. The can be indicated by passing a string in fbirth: either "rate_id" or "rate_const". The string "rate_const" corresponds to a constant rate and has to have the rate's value in field called $rate. However, a switching of rates is allowed. If the value of the first trait exceeds a certain threshold (provided in field $switch of birth parameters), then the rate is changed to the value in $rate2, see body of hidden function .f_rate_const(), in file rates.R. The string "rate_id" corresponds to the .f_rate_id() function, in file rates.R. If the birth parameters are NULL, then the rate equals the value of the first phenotype. However, a number of linear, threshold and power transformations of the rate are possible.The field varnum indicates the index of the variable to take as the one influencing the rate (remember to add 1 for the time entry). Then, if x stands for the trait influencing the branching rate it is transformed into a rate by the following fields in the following order. Set rate<-x and let params correspond to the list containing the branching parameters. • substractbaserate<-max(0,params$rate-substractbase) rate<-abs(rate) • p and if is.null(params$raise.at.end) || !params$raise.at.end rate<-rate^params$p • base and !is.null(params$const) if (res<params$base){rate<-params$const} • base and is.null(params$const) if (res<params$base){rate<-0} • invbase and !is.null(params$const) if (res>params$invbase){rate<-params$const} • invbase and is.null(params$const) if (res>params$invbase){rate<-0} • scalerate<-rate/params$scale • p and if params$raise.at.endrate<-rate^params$p res<-abs(res) • maxvalif(rate>params$maxval){rate<-params$maxval}
fdeath	A function that returns the birth rate at a given moment. Its structure is the same as fbirth. The current version of the package does not provide any support for any inbuilt function.

`fbirth.params` The parameters of the birth rate, to be provided to `fbirth`. They have to be a named list.

`fdeath.params` The parameters of the death rate, to be provided to `fdeath`. They have to be a named list.

`fsimulphenotype`

The name of a function to simulate the phenotype over a period of time. The function has to have four parameters (in the following order not by name): `time`, `params`, `X0` and `step`. The parameter `time` is the duration of the simulation, i.e. the length of the branch. The parameter `params` is a list of the parameters governing the simulation, what will be passed here is the list `phenotype.model.params`, without the fields `fixed`, `abstepsd` and `positivevars`. It is up to the function in `phenotype.model` to interpret the provided parameters. `X0` stands for the value at the start of the branch and `step` is a control parameter governing the time step size of the simulation. The **pcmabc** package has inbuilt support for simulating the trait as as stochastic differential equation by the **yuima** package with the function `simulate_sde_on_branch()`. However the user needs to write the `phenotype.model` function that translates the vector of parameters into an object that is understandable by **yuima** and then call `simulate_sde_on_branch()`, see the Example.

The phenotype is simulated prior to the simulation of the speciation/extinction events on a lineage (see Details). Hence, it is not possible (at the moment) to include some special event (e.g. a cladogenetic jump) at branching. Such dynamics are only possible at the start of the lineage, i.e. when the new lineage separated from the main lineage. Hence, cladogenetic change can only be included as an event connected with a lineage (subpopulation) breaking off.

`n.contemporary` The number of contemporary species to generate. If equals -1, then ignored. Otherwise when the tree reaches `n.contemporary` tips at height `tree.height` the simulation is stopped. However, there is no conditioning on this value, it is just an upper bound. It may just happen that due to the birth and death rate functions the process stops before reaching the target number of tips.

`n.tips.total` The total (contemporary and extinct) number of tips to generate. If equals -1, then ignored. Otherwise when the tree reaches `n.tips.total` tips the simulation is stopped. However, there is no conditioning on this value, it is just an upper bound. It may just happen that due to the birth and death rate functions the process stops before reaching the target number of tips.

`step` The time step size for simulating the phenotype. If not provided, then calculated as `min(0.001,tree.height/1000)`.

Details

The tree is simulate by means of a Cox process (i.e. Poisson process with random rate). First the trait is simulated along the spine of a tree, i.e. a lineage of duration `tree.height`. Then, along this spine the birth and death rates are calculated (they may depend on the value of the phenotype). The maximum for each rate is calculated and a homogeneous Poisson process with the maximum rate is simulated. Then, these events are thinned. Each event is retained with probability equalling true rate divided by maximum of rate (p. 32, Sheldon 2006). All speciation events are retained until the first death event.

Value

`tree` The simulated tree in `phylo` format.

`phenotype` A list of the trajectory of the simulated phenotype. The i-th entry of the list corresponds to the trait's evolution on the i-th edge (as in i-th row of `phyltree$edge`) of the tree. Each entry is a matrix with first row equalling time (relative to the start of the branch) and the next rows correspond to the trait value(s).

`root.branch.phenotype`
The simulation on the root branch. A matrix with first row being time and next rows the simulated trait(s).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

Sheldon R. M. (2006). Simulation. Elsevier Academic Press.

See Also

PCM_ABC

Examples

```
## simulate 3d OUBM model with id branching rate
set.seed(12345)

simulate_mvsl_sde<-function(time,params,X0,step){
    A <- c(paste("(-",params$a11,")*(x1-(",params$psi1,"))
    -(",params$a12,")*(x2-(",params$psi2,"))-(",params$b11,")*x3",sep=""),
    paste("(-",params$a21,")*(x1-(",params$psi1,"))
    -(",params$a22,")*(x2-(",params$psi2,"))-(",params$b21,")*x3",sep=""),0)
    S <- matrix( c( params$s11, params$s12, 0, 0, params$s22
    , 0, 0, 0, params$s33), 3, 3,byrow=TRUE)
    yuima.3d <- yuima::setModel(drift = A, diffusion = S,
    state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
    simulate_sde_on_branch(time,yuima.3d,X0,step)
}

birth.params<-list(scale=1,maxval=2)

sde.params<-list(a11=2.569531,a12=0,a21=0,a22=28.2608,b11=-5.482939,
b21=-34.806936,s11=0.5513215,s12=1.059831,s22=1.247302,s33=1.181376,
psi1=-2.4590422,psi2=-0.6197838)
X0<-c(5.723548,4.103157,3.834698)
step<-0.25 ## for keeping example's running time short <5s as CRAN policy,
           ## in reality should be much smaller e.g. step<-0.001
```

```
simres<-simulate_phylproc(3.5, sde.params, X0, fbirth="rate_id", fdeath=NULL,
fbirth.params=NULL, fdeath.params=NULL, fsimulphenotype=simulate_mvsl_sde,
n.contemporary=5, n.tips.total=-1, step=step)
```

`simulate_sde_on_branch` *Simulate a stochastic differential equation on a branch. using the* **yuima**

Description

The function simulates a stochastic differential equation on a branch using the **yuima** package.

Usage

```
simulate_sde_on_branch(branch.length, model.yuima, X0, step)
```

Arguments

`branch.length`	The length of the branch.
`model.yuima`	A object that **yuima** can understand in order to simulate a stochastic differential equation, see Example.
`X0`	The value at the start of the branch.
`step`	The simulation step size that is provided to **yuima**.

Details

The function is a wrapper for calling `yuima::simulate()`.

Value

It returns a matrix whose first row are the time points on the branch and the remaining rows the values of the trait(s).

Author(s)

Krzysztof Bartoszek

References

Bartoszek, K. and Lio', P (2019). Modelling trait dependent speciation with Approximate Bayesian Computation. Acta Physica Polonica B Proceedings Supplement 12(1):25-47.

Brouste A., Fukasawa M., Hino H., Iacus S. M., Kamatani K., Koike Y., Masuda H., Nomura R., Ogihara T., Shimuzu Y., Uchida M., Yoshida N. (2014). The YUIMA Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. Journal of Statistical Software, 57(4): 1-51.

Iacus S. M., Mercuri L., Rroji E. (2017). COGARCH(p,q): Simulation and Inference with the yuima Package. Journal of Statistical Software, 80(4): 1-49.

See Also

setModel, setSampling, simulate,

Examples

```
## simulate a 3D OUBM process on a branch
set.seed(12345)

A <-c("-(x1-1)-2*x3","-(x2+1)+2*x3",0)
S <- matrix( c( 1, 2, 0, 0, 1 , 0, 0, 0,
2), 3, 3,byrow=TRUE)
yuima.3d <- yuima::setModel(drift = A, diffusion = S,
state.variable=c("x1","x2","x3"),solve.variable=c("x1","x2","x3") )
X0<-c(0,0,0)
step<-0.5 ## for keeping example's running time short <5s as CRAN policy,
          ## in reality should be much smaller e.g. step<-0.001

time<-1
simulate_sde_on_branch(time,yuima.3d,X0,step)
```

Index