Package ‘BBMV’

October 23, 2017

Type Package

Title Models for Continuous Traits Evolving in Macroevolutionary Landscapes of any Shape

Version 2.0

Date 2017-10-23

Author Florian C. Boucher

Maintainer Florian C. Boucher `<floflobboucher@gmail.com>`

Depends R (>= 3.1.0), ape

Suggests coda, geiger

Description Provides a set of functions to fit general macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape. This package implements the Fokker-Planck-Kolmogorov model (FPK), in which the trait evolves under random diffusion but is also subject to a force that pulls it towards specific values - this force can be of any shape. FPK has a version in which hard reflective bounds exist at the extremes of the trait interval: this second model is called BBMV.

License GPL-2

NeedsCompilation no

R topics documented:

`BBMV-package` *Models for Continuous Traits Evolving in Macroevolutionary Landscapes of any Shape*

Description

Provides a set of functions to fit general macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape. This package implements the Fokker-Planck-Kolmogorov model (FPK), in which the trait evolves under random diffusion but is also subject to a force that pulls it towards specific values - this force can be of any shape. FPK has a version in which hard reflective bounds exist at the extremes of the trait interval: this second model is called BBMV.

Details

Package:	BBMV
Type:	Package
Title:	Models for Continuous Traits Evolving in Macroevolutionary Landscapes of any Shape
Version:	2.0
Date:	2017-10-23
Author:	Florian C. Boucher
Maintainer:	Florian C. Boucher <floflobaucher@gmail.com>
Depends:	R (>= 3.1.0), ape
Suggests:	coda, geiger
Description:	Provides a set of functions to fit general macroevolutionary models for continuous traits evolving in adapti
License:	GPL-2

Index of help topics:

```
ACE_FPK                 Ancestral Character Estimation
BBMV-package            Models for Continuous Traits Evolving in
                        Macroevolutionary Landscapes of any Shape
ConvProp_bounds         Convolution of the diffusion matrix with the
                        trait density vector.
DiffMat_backwards       Diffusion matrix building
DiffMat_forward         Diffusion matrix building
```

`FPK_sim_traitgram`	Simulations with traitgram
`FormatTree_bounds`	Tree formatting.
`LogLik_bounds`	Likelihood of the FPK model
`MH_MCMC_FPK`	MCMC estimation
`MH_MCMC_FPK_multiclades`	MCMC estimation on multiple clades
`Sim_FPK`	Simulation of the BBM+V process.
`Uncertainty_FPK`	Parameter uncertainty
`VectorPos_bounds`	Discretization of a continuous trait value into a probability vector.
`charac_time`	Characteristic time measurement
`find.mle_FPK`	Maximum-likelihood estimation
`find.mle_FPK_multiple_clades_same_V_different_sig2`	Maximum-likelihood estimation
`fit_FPK_multiple_clades_different_V_different_sig2`	Fit independent models in several clades.
`get.landscape.FPK`	Plot Plot macroevolutionary landscapes estimated by the FPK or BBM+V models
`get.landscape.FPK.MCMC`	Plot posterior distribution of macroevolutionary landscapes.
`lnL_FPK`	Creation of the likelihood function
`lnl_FPK_multiclades_same_V_different_sig2`	Likelihood functions for multiple clades
`log_prior_5pars_root_bounds`	Prior function.
`log_prior_nclades_plus_3_pars`	Prior function.
`prep_mat_exp`	Matrix exponential.
`proposal_5pars_root_bounds`	Parameter update for the MCMC function
`proposal_nclades_plus_3_pars`	Parameter update for the multiclade MCMC function
`reformat_multiclade_results`	Format the output of a multiclade fit
`trans_from_fixed`	Linear transformations

Author(s)

Florian C. Boucher Maintainer: Florian C. Boucher <floflobboucher@gmail.com>

References

Inferring bounded evolution in phenotypic characters from phylogenetic comparative data. F.C. Boucher and V. Demery. Systematic Biology, 65, 651-661, 2016

A general model for estimating macroevolutionary landscapes from phylogenetic comparative data. F.C. Boucher, V. Demery, E. Conti, L. J. Harmon and J. Uyeda. Systematic Biology, syx075, https://doi.org/10.1093/sysbio/syx075

`ACE_FPK` *Ancestral Character Estimation*

Description

Function to perform Ancestral Character Estimation under the FPK (or BBM+V) model

Usage

```
ACE_FPK(fit, specific.point = NULL)
```

Arguments

`fit`	An FPK model fit, as returned by find.mle_FPK.
`specific.point`	If set to NULL (the default), then the function will produce an ACE at all internal nodes in the tree. Alternatively, specific.point can be used to ask for an ACE at any specific point in the tree (i.e., not a node): specific.point must then be a vector with three elements: c(parent_node,child_node,time_from_start_of branch).

Value

For each internal node, the function returns a table giving the probability density of the trait. The first column gives all possible trait values on the discretized trait grid, and the second the probability density at each of these points. If only a specific.point was asked, the function only returns one such table.

Author(s)

F. C. Boucher

`charac_time` *Characteristic time measurement*

Description

Calculate the characteristic time it takes for the FPK process to reach its stationary distribution.

Usage

```
charac_time(Npts, fit)
```

Arguments

`Npts`	The number of points used in the discretization procedure.
`fit`	A FPK model fit, as returned by find.mle_FPK.

Value

The function returns the characteristic time of the process as a numeric value.

Author(s)

F. C. Boucher

`ConvProp_bounds` *Convolution of the diffusion matrix with the trait density vector.*

Description

Internal function used for likelihood calculation and simulation.

Usage

```
ConvProp_bounds(X, t, prep_mat)
```

Arguments

`X`	A trait density vector.
`t`	The time over which to do the convolution (usually the length of one branch).
`prep_mat`	The diagonalized diffusion matrix.

Author(s)

F. C. Boucher

`DiffMat_backwards` *Diffusion matrix building*

Description

Internal function that builds the discretized diffusion matrix of the FPK process going backwards in time (for likelihood calculations)

Usage

```
DiffMat_backwards(V)
```

Arguments

`V`	A vector giving the values of the evolutionary potential (V) at each point in the gridded trait interval.

Author(s)

F. C. Boucher

`DiffMat_forward` *Diffusion matrix building*

Description

Internal function that builds the discretized diffusion matrix of the FPK process going forward in time (for simulations)

Usage

```
DiffMat_forward(V)
```

Arguments

`V`	A vector giving the values of the evolutionary potential (V) at each point in the gridded trait interval.

Author(s)

F.C. Boucher

`find.mle_FPK` *Maximum-likelihood estimation*

Description

Find the maximum-likelihood estimate of the FPK model.

Usage

```
find.mle_FPK(model, method = "Nelder-Mead", init.optim = NULL, safe = F)
```

Arguments

`model`	An FPK or BBMV model, as generated by lnL_FPK or lnL_BBMV.
`method`	The optimization routine to be used: can be either "Nelder-Mead" (the default) or "L-BFGS-B". See the documentation of the optim function for more details. From our experience, "Nelder-Mead" seems to produce better results.
`init.optim`	A vector of initial values for model parameters to start the optimization algorithm. If left NULL (as is by default), the function chooses a reasonable starting point, but you might want to play around with it.
`safe`	If safe is set to TRUE, the function runs three different optimizations starting with different values of the rate of evolution (sigma). This can prove useful in difficult cases. Default to FALSE for a single optimization (which is quicker).

Value

A list with the following elements:

`lnL`	the log-likelihood of the model
`aic`	the Akaike Information Criterion of the model
`k`	the number of parameters of the model
`par`	a list of the MLEs of model parameters
`par_fixed`	a list with the parameters that were fixed. This includes the bounds use to discretize the model and eventually some of the parameters describing the shape of the macroevolutionary landscape.
`root`	A table giving the probability density of the trait at the root of the tree. The first column gives all possible trait values on the discretized trait grid, and the second the probability density at each of these points.
`convergence`	Convergence code returned by optim. 0 indicates successful convergence. For other values see the help of the optim function.
`message`	Convergence message returned by optim. See the help of the optim function.
`tree`	the tree used as input
`trait`	the trait vector used as input
`Npts`	the number of points used to discretize trait space.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
# they are just here for technical purposes but are not reached
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# fit the FPK model:
ll_FPK4=lnL_FPK(tree,TRAIT,Npts=25,a=NULL,b=NULL,c=NULL) # the full model
fit4=find.mle_FPK(model=ll_FPK4)

## End(Not run)
```

`find.mle_FPK_multiple_clades_same_V_different_sig2` *Maximum-likelihood estimation*

Description

Maximum-likelihood estimation of the FPK model on multiple clades at once.

Usage

```
find.mle_FPK_multiple_clades_same_V_different_sig2(model,
  method = "Nelder-Mead", init.optim = NULL)
find.mle_FPK_multiple_clades_same_V_same_sig2(model,
  method = "Nelder-Mead", init.optim = NULL)
```

Arguments

`model`	An FPK or BBMV model fitted to multiple clades, as generated by lnl_FPK_multiclades_same_V_dif lnl_BBMV_multiclades_same_V_different_sig2, lnl_BBMV_multiclades_same_V_same_sig2, or lnl_FPK_multiclades_same_V_same_sig2.
`method`	The optimization routine to be used: can be either "Nelder-Mead" (the default) or "L-BFGS-B". See the documentation of the optim function for more details. From our experience, "Nelder-Mead" seems to produce better results.
`init.optim`	A vector of initial values for model parameters to start the optimization algorithm. If left NULL (as is by default), the function chooses a reasonable starting point, but you might want to play around with it.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# We first create a potential that we will use to simulate trait evolution
# It has two peaks of very unequal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
a=8 ; b=-4 ; c=1
V6=a*x^4+b*(x^2)+c*x
step_size=(max(bounds)-min(bounds))/(100-1)
V6_norm=exp(-V6)/sum(exp(-V6)*step_size) # the step size on the grid
par(mfrow=c(1,1))
plot(V6_norm,type='l')

# Now we simulate a tree and a continuous trait for 3 independent clades.
#The trait evolves in the same macroevolutionary landscape but with different evolutionary rates.
tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=1,bounds=bounds)
tree1=tree ; TRAIT1=TRAIT
```

```
tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.5,bounds=bounds)
tree2=tree ; TRAIT2=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.1,bounds=bounds)
tree3=tree ; TRAIT3=TRAIT
rm(tree) ; rm(TRAIT)

TREES=list(tree1,tree2,tree3)
TRAITS=list(TRAIT1,TRAIT2,TRAIT3)

# Fit the FPK model using ML
testbFPK4=lnl_FPK_multiclades_same_V_different_sig2(trees=TREES,
  traits=TRAITS,a=NULL,b=NULL,c=NULL,Npts=50)
fitbFPK4=find.mle_FPK_multiple_clades_same_V_different_sig2(model=testbFPK4,
  method='Nelder-Mead',init.optim=NULL)

## End(Not run)
```

`fit_FPK_multiple_clades_different_V_different_sig2`
Fit independent models in several clades.

Description

This function is a wrapper for functions that fit the FPK or BBMV model to a single clade that does it repetitively over several clades.

Usage

```
fit_FPK_multiple_clades_different_V_different_sig2(trees, traits,
  a = NULL, b = NULL, c = NULL, Npts = 50, method = "Nelder-Mead", init.optim = NULL)
fit_BBMV_multiple_clades_different_V_different_sig2(trees, traits, bounds,
  a = NULL, b = NULL, c = NULL, Npts = 50, method = "Nelder-Mead", init.optim = NULL)
```

Arguments

`trees`	A list of phylogenetic trees in 'phylo' format, one per clade.
`traits`	A list of trait vectors for species in each clade. Should be in the same order as trees.
`bounds`	The two bounds that constrain trait values when fitting the BBMV model. Specified by a numeric vector containing the minimum and maximum bound of the trait interval as the first and second element, respectively.
`a`	The value of the x^4 term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`b`	The value of the quadratic term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.

`c`	The value of the linear term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`Npts`	The number of points used in the discretization procedure.
`method`	The optimization routine to be used: can be either "Nelder-Mead" (the default) or "L-BFGS-B". See the documentation of the optim function for more details. From our experience, "Nelder-Mead" seems to produce better results.
`init.optim`	A vector of initial values for model parameters to start the optimization algorithm. If left NULL (as is by default), the function chooses a reasonable starting point, but you might want to play around with it.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# We first create a potential that we will use to simulate trait evolution
# It has two peaks of very unequal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
a=8 ; b=-4 ; c=1
V6=a*x^4+b*(x^2)+c*x
step_size=(max(bounds)-min(bounds))/(100-1)
V6_norm=exp(-V6)/sum(exp(-V6)*step_size) # the step size on the grid
par(mfrow=c(1,1))
plot(V6_norm,type='l')

# Now we simulate a tree and a continuous trait for 3 independent clades.
tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=1,bounds=bounds)
tree1=tree ; TRAIT1=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.5,bounds=bounds)
tree2=tree ; TRAIT2=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.1,bounds=bounds)
tree3=tree ; TRAIT3=TRAIT
rm(tree) ; rm(TRAIT)

TREES=list(tree1,tree2,tree3)
TRAITS=list(TRAIT1,TRAIT2,TRAIT3)

# Fit the FPK model using ML
fitmFPK4=fit_FPK_multiple_clades_different_V_different_sig2(trees=TREES,
  traits=TRAITS,a=NULL,b=NULL,c=NULL,Npts=50)


## End(Not run)
```

`FormatTree_bounds` *Tree formatting.*

Description

Internal function used for likelihood calculation and simulation.

Usage

```
FormatTree_bounds(tree, trait, V, bounds)
```

Arguments

`tree`	A phylogenetic tree in phylo format.
`trait`	A vector of traits at the tips of the tree or a list with vectors of multiple measurements for each tip.
`V`	A vector giving the evolutionary potential.
`bounds`	A vector of bounds of the trait interval.

Author(s)

F. C. Boucher

`FPK_sim_traitgram` *Simulations with traitgram*

Description

Simulates a trait evolving according to the FPK model and also plots evolution along branches of this tree using a traitgram.

Usage

```
FPK_sim_traitgram(tree, x0, a, b, c, bounds, sigsq, time_step, res.x = 200
  , ylim.plot = NULL, return.trait = FALSE)
```

Arguments

`tree`	The phylogenetic tree on which to simulate the trait evolving.
`x0`	The initial value of the trait (at the root of the tree).
`a`	The coefficient for the x^4 term of the potential.
`b`	The coefficient for the quadratic term of the potential.
`c`	The coefficient for the linear term of the potential.
`bounds`	The bounds on the trait interval.
`sigsq`	The evolutionary rate.
`time_step`	The time step for incremental simulations (should be smaller that the shortest branch in the tree).

`res.x`	The number of points to use for discretizing the trait interval.
`ylim.plot`	The y limits of the plot. If left to NULL, the bounds of the trait interval will be used, but you might want to zoom in a bit more if the bounds are not reached.
`return.trait`	If set to TRUE, the function returns a named vector of trait values at the tips of the tree.

Details

The function is slower than Sim_FPK since it simulates step by step along branches. It should be used to visualize trait evolution, but Sim_FPK should be preferred for quick simulations.

Value

The function is mainly designed to produce a plot, but eventually returns a named vector of trait values at the tips of the tree.

Author(s)

F. C. Boucher

See Also

`Sim_FPK`

`get.landscape.FPK` *Plot Plot macroevolutionary landscapes estimated by the FPK or BBM+V models*

Description

Plot the adaptive landscape estimated in a BBM+V model.

Usage

```
get.landscape.FPK(fit, Npts = 100, main = "Macroevolutionary landscape"
  , ylab = "N.exp(-V)", xlab = "Trait", xlim = NULL, ylim = NULL)
```

Arguments

`fit`	An FPK model fit, as returned by find.mle_FPK.
`Npts`	The number of points used to discretize the trait interval for plotting.
`main`	Title for the plot.
`ylim`	The upper limit of the plotting region when multiple adaptive landscapes are plotted together.
`xlim`	The limits of thex-axis.
`ylab`	Label of the y-axis.
`xlab`	Label of the x-axis.

Value

A plot of the adaptive landscape across the trait interval.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# fit the FPK model:
ll_FPK4=lnL_FPK(tree,TRAIT,Npts=25,a=NULL,b=NULL,c=NULL) # the full model
fit4=find.mle_FPK(model=ll_FPK4)
# Plot the landscape estimated
get.landscape.FPK(fit=fit4)

## End(Not run)
```

`get.landscape.FPK.MCMC` *Plot posterior distribution of macroevolutionary landscapes.*

Description

The function plots the median value of the macroevolutionary landscape across the posterior in a solid line and draws a polygon that streches between two quantiles of the posterior.

Usage

```
get.landscape.FPK.MCMC(chain, bounds, Npts = 100, burnin = 0.1,
  probs.CI = c(0.05, 0.95), COLOR_MEDIAN = "red", COLOR_FILL = "red",
  transparency = 0.3, main = "Macroevolutionary landscapes MCMC",
  ylab = "N.exp(-V)", xlab = "Trait", xlim = NULL, ylim = NULL)
```

Arguments

`chain`	An data.frame object representing the output of an MCMC chain, as obtained by MH_MCMC_FPK.
`bounds`	The bounds on the trait interval
`Npts`	The number of points used in the discretization procedure.
`burnin`	The percentage of generations discarded as burnin.
`probs.CI`	A vector of the two quantiles of the posterior distribution between which samples should be considered.
`COLOR_MEDIAN`	The color used to plot the median macroevolutionary landscape across the posterior.

COLOR_FILL	The color used to plot the polygon that stretches between the two quantiles of the posterior.
transparency	The transparency used for plotting the polygon
main	Title of the graph.
ylab	y label of the graph.
xlab	X label of the graph.
xlim	
ylim	

Author(s)

F.C. Boucher

See Also

MH_MCMC_FPK

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating: for technical purposes only
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# Run a MCMC chain to fit the FPK model
MCMC=MH_MCMC_FPK(tree,trait=TRAIT,bounds=c(5,5),Nsteps=10000,record_every=100,
  plot_every=100,Npts=20,pars_init=c(0,-4,-4,0,1),prob_update=c(0.2,0.25,0.25,0.25,0.05),
   verbose=TRUE,plot=TRUE,save_to='MCMC_FPK_test.Rdata',save_every=100,
   type_priors=c(rep('Normal',4),'Uniform'),
   shape_priors=list(c(0,10),c(0,10),c(0,10),c(0,10),NA),proposal_type='Uniform',
   proposal_sensitivity=c(0.1,0.1,0.1,0.1,1),prior.only=F)
get.landscape.FPK.MCMC(chain=MCMC,bounds=c(5,5),Npts=100,burnin=0.1,
   probs.CI=c(0.025,0.975),COLOR_MEDIAN='red',COLOR_FILL='red',transparency=0.3,
   main='Macroevolutionary landscapes MCMC',ylab='N.exp(-V)',xlab='Trait',
   xlim=NULL,ylim=NULL)

## End(Not run)
```

---

lnL_FPK *Creation of the likelihood function*

Description

Functions that builds the likelihood function of the FPK or BBMV model

Usage

```
lnL_FPK(tree, trait, a = NULL, b = NULL, c = NULL, Npts)
lnL_BBMV(tree, trait,bounds, a = NULL, b = NULL, c = NULL, Npts)
```

Arguments

`tree`	A phylogenetic tree in 'phylo' format
`trait`	A named vector of trait values for the tips of the tree. It should match tip labels in the phylogeny. Alternatively, a named list with one element per tip in the tree, each element being in turn a numeric vector with multiple measurements of the trait for this tip.
`bounds`	The two bounds that constrain trait values when fitting the BBMV model. Specified by a numeric vector containing the minimum and maximum bound of the trait interval as the first and second element, respectively.
`a`	The value of the x^4 term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`b`	The value of the quadratic term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`c`	The value of the linear term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`Npts`	The number of points used in the discretization procedure.

Value

A list of several items, including the data and model call, but most importantly the likelihood function ($fun element).

Author(s)

F.C. Boucher

See Also

`find.mle_FPK`

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# create a likelihood function for the FPK model:
```

```
ll_FPK4=lnL_FPK(tree,TRAIT,Npts=25,a=NULL,b=NULL,c=NULL) # the full model

## End(Not run)
```

---

lnl_FPK_multiclades_same_V_different_sig2

Likelihood functions for multiple clades

Description

These functions create likelihood functions of the FPK or BBMV model over multiple, independent, clades.

Usage

```
lnl_FPK_multiclades_same_V_different_sig2(trees, traits,
  a = NULL, b = NULL, c = NULL, Npts = 50)
lnl_FPK_multiclades_same_V_same_sig2(trees, traits,
  a = NULL, b = NULL, c = NULL, Npts = 50)
lnl_BBMV_multiclades_same_V_different_sig2(trees, traits,bounds,
  a = NULL, b = NULL, c = NULL, Npts = 50)
lnl_BBMV_multiclades_same_V_same_sig2(trees, traits,bounds,
  a = NULL, b = NULL, c = NULL, Npts = 50)
```

Arguments

`trees`	A list of phylogenetic trees in 'phylo' format, one per clade.
`traits`	A list of trait vectors for species in each clade. Should be in the same order as trees.
`bounds`	The two bounds that constrain trait values when fitting the BBMV model. Specified by a numeric vector containing the minimum and maximum bound of the trait interval as the first and second element, respectively.
`a`	The value of the x^4 term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`b`	The value of the quadratic term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`c`	The value of the linear term in the evolutionary potential. If set to NULL (the default), this parameter will be estimated. If a numeric value is provided, this parameter will be fixed to the value specified.
`Npts`	The number of points used in the discretization procedure.

Author(s)

F. C. Boucher

See Also

`find.mle_FPK_multiple_clades_same_V_different_sig2` `find.mle_FPK_multiple_clades_same_V_same_sig2`

Examples

```
## Not run:
# We first create a potential that we will use to simulate trait evolution
# It has two peaks of very unequal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
a=8 ; b=-4 ; c=1
V6=a*x^4+b*(x^2)+c*x
step_size=(max(bounds)-min(bounds))/(100-1)
V6_norm=exp(-V6)/sum(exp(-V6)*step_size) # the step size on the grid
par(mfrow=c(1,1))
plot(V6_norm,type='l')

# Now we simulate a tree and a continuous trait for 3 independent clades.
tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=1,bounds=bounds)
tree1=tree ; TRAIT1=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.5,bounds=bounds)
tree2=tree ; TRAIT2=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.1,bounds=bounds)
tree3=tree ; TRAIT3=TRAIT
rm(tree) ; rm(TRAIT)

TREES=list(tree1,tree2,tree3)
TRAITS=list(TRAIT1,TRAIT2,TRAIT3)

# Fit the FPK model using ML
testbFPK4=lnl_FPK_multiclades_same_V_different_sig2(trees=TREES,
  traits=TRAITS,a=NULL,b=NULL,c=NULL,Npts=50)
fitbFPK4=find.mle_FPK_multiple_clades_same_V_different_sig2(model=testbFPK4,
  method='Nelder-Mead',init.optim=NULL)

## End(Not run)
```

`LogLik_bounds` *Likelihood of the FPK model*

Description

Internal functions use to calculate the likelihood of the FPK model, used in ML and MCMC estimation.

Usage

```
LogLik_bounds(tree_formatted, dCoeff, dMat, bounds)
LogLik_bounds_est_root(tree, trait, dCoeff, V, x0_pos, bounds)
```

Arguments

`tree_formatted`	A formatted tree as returned by FormatTree_bounds.
`dCoeff`	The diffusion coefficient.
`dMat`	The discretized diffusion matrix.
`V`	A numeric vector giving the value of the evolutionary potential in each point of the trait grid.
`bounds`	A vector giving the bounds of the trait interval.
`x0_pos`	The value of the trait at the root of the tree.
`tree`	A phylogenetic tree in phylo format.
`trait`	A trait vector for tip taxa.

Author(s)

F. C. Boucher

`log_prior_5pars_root_bounds`
Prior function.

Description

Internal function that calculates the log prior, used in MCMC estimation of the BBM+V model.

Usage

```
log_prior_5pars_root_bounds(type = c(rep("Normal", 4), "Uniform") , shape =
  list(c(0, 10), c(0, 10), c(0, 10), c(0, 10), NA), pars, Npts)
```

Arguments

`type`	A vector giving the type of prior for each parameter.
`shape`	A list giving the shape of the prior for each parameter.
`pars`	The parameter values at which the prior should be evaluated.
`Npts`	The number of points on the grid.

Author(s)

F. C. Boucher

`log_prior_nclades_plus_3_pars` *Prior function.*

Description

Internal function that calculates the log prior, used in MCMC estimation of the BBM+V model.

Usage

```
log_prior_nclades_plus_3_pars(type = NULL, shape = NULL, pars, n_clades)
```

Arguments

`type`	A vector giving the type of prior for each parameter.
`shape`	A list giving the shape of the prior for each parameter.
`pars`	The parameter values at which the prior should be evaluated.
`n_clades`	The number of clades included in the multiclade analysis.

Author(s)

F. C. Boucher

`MH_MCMC_FPK` *MCMC estimation*

Description

The function estimates the parameters of the BBM+V model using an MCMC chain with the Metropolis Hastings algorithm and a Gibbs sampler.

Usage

```
MH_MCMC_FPK(tree, trait, bounds, Nsteps = 5e+05, record_every = 100, plot_every = 500,
  Npts = 50, pars_init = c(0, 0, 0, 0, 25), prob_update = c(0.2, 0.2, 0.2, 0.2, 0.2),
  verbose = TRUE, plot = TRUE, save_to = "MCMC_FPK_test.Rdata", save_every = 10000,
   type_priors = c(rep("Normal", 4), "Uniform"),
   shape_priors = list(c(0, 10), c(0, 10), c(0, 10), c(0, 10), NA),
   proposal_type = "Uniform", proposal_sensitivity = c(0.1, 0.1, 0.1, 0.1, 1),
   prior.only = F, burnin.plot = 0.1)
```

Arguments

`tree`	A phylogenetic tree in phylo format.
`trait`	A named vector of trait values for the tips of the tree. It should match tip labels in the phylogeny.
`bounds`	A vector with two elements giving the bounds on the trait interval.
`Nsteps`	The number of generations in the MCMC chain.
`record_every`	The frequency used for sampling the MCMC chain.
`plot_every`	The frequency at which the chain is plotted (if plot=TRUE).
`Npts`	The number of points on the grid between the bounds.
`pars_init`	A vector giving the initial parameters for starting the algorithm, which correspond to the following: c(log(sig^2/2),a,b,c,x0).
`prob_update`	A vector giving the relative frequencies of update of the different parameters of the model.
`verbose`	If TRUE, will print some generations of the chain to the screen.
`plot`	If TRUE, the chain is plotted from time to time.
`save_to`	The path to the file where the chain is saved (can be useful in case the chain crashes).
`save_every`	Sets how often the chain is saved.
`type_priors`	A character vector specifying the type of priors used. Either 'Uniform' or 'Normal'. See Details.
`shape_priors`	A list that gives the shape for each prior. See Details.
`proposal_type`	The type of proposal function, only 'Uniform' is available (the default).
`proposal_sensitivity`	A numeric vector specifying the width of the uniform proposal for each parameter. See Details.
`prior.only`	Default to FALSE for estimation of the posterior. If TRUE, the likelihood is not evaluated: this is mostly useful for internal test of the Gibbs sampler.
`burnin.plot`	The percentage of samples considered as burnin and thus not shown on the trace plot that the function produces.

Details

When specifying intial parameters yourself, be careful since x0 is actually the index of the point on the grid (between 1 and Npts_int), not the actual root value. Also the first parameter is the diffusion coefficient (log(sig^2/2)), not the evolutionary rate (sig^2). Finally, be careful that the bounds you propose must contain all trait values in you dataset.

Priors can be either 'Normal' (preferred) or 'Uniform' for log(sig^2/2), a, b and c. The only option for x0 is a discrete uniform prior, specified by 'Uniform'.

Each element of the shape_priors list should be a vector giving c(mean,sd) for normal priors and c(min,max) for continuous uniform priors. The shape is not specified for the root prior (it is set as 'NA' by default), since it is fixed to be discrete uniform on the grid.

Elements of the proposal_sensitivity vector can be any positive number for continuously varying parameters: c(log(sig^2/2),a,b,c). Default values should often be a good start. Only integer numbers are possible for x0 and give how many steps at a time can be travelled on the trait grid when updating these parameters. It is recommended to keep it to 1, as it is by default.

Value

A matrix of numeric values giving values of all parameters, the likelihood, prior and posterior at each generation sampled in the MCMC chain (one row per sample taken). The matrix has the following columns:

`step`	The number of the generation sampled.
`sigsq`	The evolutionary rate.
`a`	The coefficient of the x^4 term of the evolutionary potential.
`b`	The coefficient of the x^2 term of the evolutionary potential.
`c`	The coefficient of the x term of the evolutionary potential.
`root`	The value of the trait at the root of the tree.
`lnprior`	The logarithm of the prior.
`lnlik`	The logarithm of the likelihood.
`quasi-lnpost`	The logarithm of the (unnormalized) posterior.
`Acceptance`	Whether the proposed MCMC move was accepted (1) or not (0).
`Par_updated`	Which parameter was updated in this generation.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating: for technical purposes only
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# Run a MCMC chain to fit the FPK model
MCMC=MH_MCMC_FPK(tree,trait=TRAIT,bounds=c(5,5),Nsteps=10000,record_every=100,
  plot_every=100,Npts=20,pars_init=c(0,-4,-4,0,1),prob_update=c(0.2,0.25,0.25,0.25,0.05),
   verbose=TRUE,plot=TRUE,save_to='MCMC_FPK_test.Rdata',save_every=100,
   type_priors=c(rep('Normal',4),'Uniform'),
   shape_priors=list(c(0,10),c(0,10),c(0,10),c(0,10),NA),proposal_type='Uniform',
   proposal_sensitivity=c(0.1,0.1,0.1,0.1,1),prior.only=F)

## End(Not run)
```

MH_MCMC_FPK_multiclades

MCMC estimation on multiple clades

Description

This function estimates parameter of the FPK model on multiple clades at once, making the assumption that they share the same macroevolutionary landscape but have different rates of evolution.

Usage

```
MH_MCMC_FPK_multiclades(trees, traits, bounds, Nsteps = 5e+05, record_every = 100,
   plot_every = 500, Npts = 50, pars_init = NULL, prob_update = NULL, verbose = TRUE,
   plot = TRUE, save_to = "MCMC_FPK_test.Rdata", save_every = 10000, type_priors = NULL,
    shape_priors = NULL, proposal_type = "Normal", proposal_sensitivity = NULL,
    prior.only = F, burnin.plot = 0.1)
```

Arguments

`trees`	A list of phylogenetic trees in 'phylo' format, one per clade.
`traits`	A list of trait vectors for species in each clade. Should be in the same order as trees.
`bounds`	The two bounds that constrain trait values when fitting the BBMV model. Specified by a numeric vector containing the minimum and maximum bound of the trait interval as the first and second element, respectively.
`Nsteps`	The number of steps in the MCMC chain.
`record_every`	How often to record a generation in the chain.
`plot_every`	How often to plot the trace of the chain.
`Npts`	The number of points used in the discretization procedure.
`pars_init`	The initial parameter values.
`prob_update`	A numeric vector with the relative probability of update of each parameter of the model.
`verbose`	If TRUE, prints generations to the screen.
`plot`	If TRUE, plots the trace of parameter values along iterations during the MCMC run.
`save_to`	The directory in which the chain should be saved.
`save_every`	How often to save the chain.
`type_priors`	The type of priors used, can be either normal (preferred) or uniform for log(sig2/2), a, b and c, ; and can only be discrete uniform for x0.
`shape_priors`	A list that gives the shape for each prior. (mean,sd) for normal priors and (min,max) for continuous uniform priors. The shape is not specified for the root prior, since it is fixed to be discrete uniform on the grid.
`proposal_type`	The type of proposal function, only uniform is available so far.
`proposal_sensitivity`	The width of the uniform proposal. The entire value for x0 gives how many steps at a time can be travelled on the trait grid (better to keep it to 1)
`prior.only`	If TRUE, only the prior is explored but the likelihood is ignored. Default to false for estimation of the posterior.
`burnin.plot`	The frequency of samples that should be discarded as burnin in trace plots.

Details

When specifying intial parameters yourself, be careful since x0 is actually the index of the point on the grid (between 1 and Npts_int), not the actual root value. Also the first n parameter, n being the number of clades studied, are diffusion coefficients (log(sig^2/2)), not evolutionary rates (sig^2). Finally, be careful that the bounds you propose must contain all trait values in you dataset.

Priors can be either 'Normal' (preferred) or 'Uniform' for log(sig^2/2), a, b and c. The only option for x0 is a discrete uniform prior, specified by 'Uniform'.

Each element of the shape_priors list should be a vector giving c(mean,sd) for normal priors and c(min,max) for continuous uniform priors. The shape is not specified for the root prior (it is set as 'NA' by default), since it is fixed to be discrete uniform on the grid.

Elements of the proposal_sensitivity vector can be any positive number for continuously varying parameters: c(log(sig^2/2),a,b,c). Default values should often be a good start. Only integer numbers are possible for x0 and give how many steps at a time can be travelled on the trait grid when updating these parameters. It is recommended to keep it to 1, as it is by default.

Value

A matrix of numeric values giving values of all parameters, the likelihood, prior and posterior at each generation sampled in the MCMC chain (one row per sample taken). The matrix has the following columns:

`step`	The number of the generation sampled.
`sigsq_clade_i`	The evolutionary rate, one column per clade.
`a`	The coefficient of the x^4 term of the evolutionary potential.
`b`	The coefficient of the x^2 term of the evolutionary potential.
`c`	The coefficient of the x term of the evolutionary potential.
`lnprior`	The logarithm of the prior.
`lnlik`	The logarithm of the likelihood.
`quasi-lnpost`	The logarithm of the (unnormalized) posterior.
`Acceptance`	Whether the proposed MCMC move was accepted (1) or not (0).
`Par_updated`	Which parameter was updated in this generation.

Author(s)

F. C. Boucher

Examples

```
## Not run:
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
a=8 ; b=-4 ; c=1
V6=a*x^4+b*(x^2)+c*x
step_size=(max(bounds)-min(bounds))/(100-1)
V6_norm=exp(-V6)/sum(exp(-V6)*step_size)
par(mfrow=c(1,1))
plot(V6_norm,type='l')

# Now we simulate a tree and a continuous trait for 3 independent clades.
tree=sim.bdtree(stop='taxa',n=25) # tree with few tips for quick tests
```

```
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=1,bounds=bounds)
tree1=tree ; TRAIT1=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.5,bounds=bounds)
tree2=tree ; TRAIT2=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.1,bounds=bounds)
tree3=tree ; TRAIT3=TRAIT
rm(tree) ; rm(TRAIT)

TREES=list(tree1,tree2,tree3)
TRAITS=list(TRAIT1,TRAIT2,TRAIT3)

# Fit the FPK model using ML:
# In all clades the macroevolutionary landscape is the same
#but they have different evolutionary rates
testbFPK4=lnl_FPK_multiclades_same_V_different_sig2(trees=TREES,
  traits=TRAITS,a=NULL,b=NULL,c=NULL,Npts=50)
fitbFPK4=find.mle_FPK_multiple_clades_same_V_different_sig2(model=testbFPK4,
  method='Nelder-Mead',init.optim=NULL)

# And now MCMC run
mcmc1=MH_MCMC_FPK_multiclades(trees=TREES,traits=TRAITS,
  bounds=fitmFPK4_SE$fits$fit_clade_1$par_fixed$bounds,Nsteps=10000,record_every=100,
  plot_every=200,Npts=25,pars_init=NULL,prob_update=NULL,verbose=TRUE,plot=TRUE,
  save_to='MCMC_FPK_test.Rdata',save_every=1000,type_priors=NULL,shape_priors=NULL,
  proposal_type='Normal',proposal_sensitivity=NULL,prior.only=F,burnin.plot=0.1)

get.landscape.FPK.MCMC(chain=mcmc1,bounds,Npts=100,burnin=0.1,probs.CI=c(0.25,0.75),
 COLOR_MEDIAN='red',COLOR_FILL='red',transparency=0.3,main='Macroevolutionary landscapes MCMC',
  ylab='N.exp(-V)',xlab='Trait',xlim=NULL,ylim=c(0,2))

## End(Not run)
```

---

prep_mat_exp *Matrix exponential.*

Description

Internal function used for likelihood calculation and simulation.

Usage

```
prep_mat_exp(dCoeff, dMat, bounds)
```

Arguments

dCoeff	The diffusion coefficient.
dMat	The diffusion matrix.
bounds	A vector with two bounds for the trait interval.

Author(s)

F. C. Boucher

---

proposal_5pars_root_bounds

Parameter update for the MCMC function

Description

Internal function that proposes parameter updates used in MCMC estimation of the BBMV model.

Usage

```
proposal_5pars_root_bounds(type = "Uniform", sensitivity, pars)
```

Arguments

`type`	The type of proposal function, only 'Uniform' is available (the default).
`sensitivity`	A numeric vector specifying the width of the uniform proposal for each parameter.
`pars`	The current parameters in the MCMC chain.

Author(s)

F. C. Boucher

---

proposal_nclades_plus_3_pars

Parameter update for the multiclade MCMC function

Description

Internal function that proposes parameter updates used in MCMC estimation of the BBMV model.

Usage

```
proposal_nclades_plus_3_pars(type = "Uniform", sensitivity, pars, n_clades)
```

Arguments

`type`	The type of proposal function, only 'Uniform' is available (the default).
`sensitivity`	A numeric vector specifying the width of the uniform proposal for each parameter.
`pars`	The current parameters in the MCMC chain.
`n_clades`	The number of clades under study.

Author(s)

F. C. Boucher

`reformat_multiclade_results` *Format the output of a multiclade fit*

Description

This functions takes the output of a multiclade fit and formats it as a list of model fits for each clade. Functions used to analyze single clade fits can then be used.

Usage

```
reformat_multiclade_results(fit)
```

Arguments

`fit`	A multiclade model fit, as returned by find.mle_FPK_multiple_clades_same_V_different_sig2, find.mle_FPK_multiple_clades_same_V_same_sig2, fit_BBMV_multiple_clades_different_V_differ or fit_FPK_multiple_clades_different_V_different_sig2.

Value

A list containing model fits for each clade, in the same format as the object returned by find.mle_FPK.

Author(s)

F.C. Boucher

Examples

```
## Not run:
# We first create a potential that we will use to simulate trait evolution
# It has two peaks of very unequal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating
a=8 ; b=-4 ; c=1
V6=a*x^4+b*(x^2)+c*x
step_size=(max(bounds)-min(bounds))/(100-1)
V6_norm=exp(-V6)/sum(exp(-V6)*step_size) # the step size on the grid
par(mfrow=c(1,1))
plot(V6_norm,type='l')

# Now we simulate a tree and a continuous trait for 3 independent clades.
tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=1,bounds=bounds)
tree1=tree ; TRAIT1=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.5,bounds=bounds)
tree2=tree ; TRAIT2=TRAIT

tree=sim.bdtree(stop='taxa',n=25)
```

```
tree$edge.length=100*tree$edge.length/max(branching.times(tree))
TRAIT= Sim_FPK(tree,x0=0.5,V=V6,sigma=0.1,bounds=bounds)
tree3=tree ; TRAIT3=TRAIT
rm(tree) ; rm(TRAIT)

TREES=list(tree1,tree2,tree3)
TRAITS=list(TRAIT1,TRAIT2,TRAIT3)

# Fit the FPK model
testbFPK4=lnl_FPK_multiclades_same_V_different_sig2(trees=TREES,
  traits=TRAITS,a=NULL,b=NULL,c=NULL,Npts=50)
fitbFPK4=find.mle_FPK_multiple_clades_same_V_different_sig2(model=testbFPK4,
  method='Nelder-Mead',init.optim=NULL)
fits=reformat_multiclade_results(fitbFPK4)

## End(Not run)
```

Sim_FPK *Simulation of the BBM+V process.*

Description

The function simulates a continuous trait evolving according to the FPK process along the branches of a phylogenetic tree.

Usage

```
Sim_FPK(tree, x0 = 0, V = rep(0, 100), sigma, bounds)
```

Arguments

`tree`	A phylogenetic tree in phylo format.
`x0`	The value of the trait at the root of the tree.
`V`	A vector giving the values of the evolutionary potential at each point of the discretized trait grid. Default is a flat potential, i.e. bounded Brownian Motion.
`sigma`	The square root of the diffusion rate.
`bounds`	A vector giving the values of the bounds of the trait interval.

Value

A numeric vector with values of the trait at the tips of the tree. Names correspond to tip labels in the tree.

Author(s)

F. C. Boucher

Examples

```
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=20) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate a trait evolving on the tree with a linear trend towards small values
TRAIT= Sim_FPK(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5))
```

`trans_from_fixed` *Linear transformations*

Description

Internal function used to convert back and forth between the actual trait interval and [-1.5;1.5]

Usage

```
trans_from_fixed(x, bounds)
trans_to_fixed(x, bounds)
```

Arguments

`x`	A single value or vector of trait values.
`bounds`	The actual bounds on the trait interval

Value

A single value or vector of trait values transformed to the other interval.

Author(s)

F. C. Boucher

`Uncertainty_FPK` *Parameter uncertainty*

Description

This function plots likelihood profiles around the MLEs of paramaters and returns 95% confidence intervals.

Usage

```
Uncertainty_FPK(fit, tree, trait, Npts = 50, effort_uncertainty = 100,
  scope_a = c(-10, 10), scope_b = c(-10, 10), scope_c = c(-10, 10))
```

Arguments

`fit`	An FPK model fit, as returned by find.mle_FPK.
`tree`	The phylogenetic tree.
`trait`	The named trait vector
`Npts`	The number of points used to discretize the trait interval.
`effort_uncertainty`	Determines the number of values at which the likelihood should be calculated for each parameter.
`scope_a`	Extreme values that should be investigated for parameter a.
`scope_b`	Extreme values that should be investigated for parameter b.
`scope_c`	Extreme values that should be investigated for parameter c.

Value

A list with 95% confidence intervals for all parameters.

Author(s)

F. C. Boucher

Examples

```
## Not run:
# Simulate data: tree + continuous trait
library(geiger)
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
tree$edge.length=100*tree$edge.length/max(branching.times(tree)) # rescale the tree
# Simulate trait evolving on a macroevolutionary landscape with two peaks of equal heights
x=seq(from=-1.5,to=1.5,length.out=100)
bounds=c(min(x),max(x)) # the bounds we use for simulating: for technical purposes only
V6=10*(x^4-0.5*(x^2)+0.*x) # this is the evolutionary potential: it has two wells
TRAIT= Sim_FPK(tree,x0=0,V=V6,sigma=10,bounds=c(-5, 5))
# fit the FPK model:
ll_FPK4=lnL_FPK(tree,TRAIT,Npts=25,a=NULL,b=NULL,c=NULL) # the full model
fit4=find.mle_FPK(model=ll_FPK4)
# Measure uncertainty on model parameters
Uncertainty_FPK(fit=fit4,tree,trait=TRAIT,Npts=25,effort_uncertainty= 100,
  scope_a=c(-1,10),scope_b=c(-5,5),scope_c=c(-2,2))

## End(Not run)
```

`VectorPos_bounds` *Discretization of a continuous trait value into a probability vector.*

Description

Internal function used for likelihood calculation and simulation.

Usage

```
VectorPos_bounds(x, V, bounds)
```

Arguments

`x`	A numeric value of the trait or a vector containing multiple measurements.
`V`	The evolutionary potential used
`bounds`	A vector with the values of both bounds.

Author(s)

F. C. Boucher

Index