Package 'BBMV'

August 4, 2016

Type Package

Title Macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape

Version 1.0

Date 2016-08-04

Author Florian C. Boucher

Maintainer Florian C. Boucher <floflobоucher@gmail.com>

Depends R (>= 3.1.0), ape

Suggests geiger, coda

Description This package provides a set of functions to fit general macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape. The model is based on bounded Brownian motion (BBM), in which a continuous trait evolves along a phylogenetic tree under constant-rate diffusion between two reflective bounds. In addition to this random component, the trait evolves in a potential and is thus subject to a force that pulls it towards specific values - this force can be of any shape. We label this model BBM+V. The package implements both maximum likelihood and MCMC estimation of model parameters. The package also contains functions to simulate traits evolving under BBM+V and plot adaptive landscapes.

License GPL-2

NeedsCompilation no

R topics documented:

`BBMV-package` *Macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape*

Description

This package provides a set of functions to fit general macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape. The model is based on bounded Brownian motion (BBM), in which a continuous trait evolves along a phylogenetic tree under constant-rate diffusion between two reflective bounds. In addition to this random component, the trait evolves in a potential and is thus subject to a force that pulls it towards specific values - this force can be of any shape. We label this model BBM+V. The package implements both maximum likelihood and MCMC estimation of model parameters. The package also contains functions to simulate traits evolving under BBM+V and plot adaptive landscapes.

Details

Package:	BBMV
Type:	Package
Title:	Macroevolutionary models for continuous traits evolving in adaptive landscapes of any shape
Version:	1.0
Date:	2016-08-04
Author:	Florian C. Boucher
Maintainer:	Florian C. Boucher <floflobouchер@gmail.com>
Depends:	R (>= 3.1.0), ape
Suggests:	geiger, coda
Description:	This package provides a set of functions to fit general macroevolutionary models for continuous traits evol
License:	GPL-2

Index of help topics:

```
BBMV-package            Macroevolutionary models for continuous traits
                        evolving in adaptive 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
FormatTree_bounds       Tree formatting.
LogLik_bounds           Likelihood calculations for the BBM+V model
MH_MCMC_V_ax4bx2cx_root_bounds
                        MCMC estimation
Optim_bBM_0_flex_pts_multiple_starts
                        Maximum-likelihood estimation
Sim_BBMV                Simulation of the BBM+V process.
```

```
VectorPos_bounds        Discretization of a continuous trait value into
                        a probability vector.
charac_time             Characteristic time measurement
fit_BBMV                BBM+V model fitting
log_prior_7pars_root_bounds
                        Prior function.
plot.landscape.BBMV     Plot adaptive landscapes
prep_mat_exp            Matrix exponential.
proposal_7pars_root_bounds
                        Parameter update
```

Author(s)

Florian C. Boucher

Maintainer: Florian C. Boucher <floflo boucher@gmail.com>

References

Inferring bounded evolution in phenotypic characters from phylogenetic comparative data. F.C. Boucher & V. Demery. Systematic biology, syw015 (Early online)

A general model for estimating adaptive landscapes of any shape from phylogenetic comparative data. F.C. Boucher, V. Demery, E. Conti, J. Uyeda & L. J. Harmon. In preparation

Examples

```
# Simulate data: tree + continuous trait
library(geiger) # we will use geiger for simulating the tree
tree=sim.bdtree(stop='taxa',n=10) # tree with few tips for quick tests
# rescale the tree to a total depth of 100
tree$edge.length=100*tree$edge.length/max(branching.times(tree))

# The simulation function
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5))

# fit a model with a linear potential using maximum-likelihood: multiple starting points in the optimization t
BBM_x=fit_BBMV(tree,TRAIT,Npts=20,method='Nelder-Mead',verbose=TRUE,V_shape='linear')

# MCMC estimation: only 20,000 generations for a quick test
MCMC= MH_MCMC_V_ax4bx2cx_root_bounds(tree,trait=TRAIT,Nsteps=2000,record_every=100,plot_every=500,Npts_int=
```

---

charac_time *Characteristic time measurement*

Description

Calculate the characteristic time it takes for the BBM+V process to reach its stationary distribution.

Usage

```
charac_time(Npts, model)
```

Arguments

Npts	The number of points used in the discretization procedure.
model	A BBM+V model fit, as returned by fit_BBMV.

Value

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

Author(s)

F. C. Boucher

Examples

```
# Simulate data: tree + continuous trait
library(geiger) # we will use geiger for simulating the tree
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 to a total depth of 100
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5)) # TRAIT simulated on th

# fit a model with a linear  potential: multiple starting points in the optimization to ensure convergence
BBM_x=fit_BBMV(tree,TRAIT,Npts=20,method='Nelder-Mead',verbose=T,V_shape='linear')

# measure time to reach stationarity
charac_time(Npts=20, BBM_x)

# compare it with tree depth
max(branching.times(tree))
```

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 BBM+V 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 BBM+V 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

`fit_BBMV` *BBM+V model fitting*

Description

Fits various forms of the BBM+V model to trait data and a phylogeny, using maximum likelihood.

Usage

```
fit_BBMV(tree, trait, Npts = 50, method = "Nelder-Mead", verbose = T, V_shape)
```

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.
`Npts`	The number of points used in the discretization procedure.
`method`	The optimization routine used.
`verbose`	If TRUE, prints progress to the screen.
`V_shape`	The shape of the evolutionary potential. This should be one of 'flat','linear','quadratic', or 'full'.

Details

Only 'Nelder-Mead' (the default, which we found to be more reliable) and 'L-BFGS-B' are supported as optimization methods. For further details on both methods see the help of the optim function.

The different values for V_shape refer to the following shapes of the evolutionary potential: (1) 'flat' is a flat potential as in BBM, i.e. $V(x) = 0$; (2) 'linear' is a linear potential, i.e. $V(x) = c.x$; (3) 'quadratic' is a quadratic potential, i.e. $V(x) = b.x^2 + c.x$; (4) 'full' is a potential with three polynomial terms, i.e. $V(x) = a.x^4 + b.x^2 + c.x$.

Value

A list with the following elements:

`par`	The ML estimates of model parameters: the two bounds of the trait interval ($bounds), the diffusion rate ($sigsq), the coefficients of the evolutionary potential ($a, $b, and $c, depending on the shape fitted), and the value of the trait at the root of the tree ($root_value).
`lnL`	The log-likelihood of the model.
`k`	The number of parameters of the model.
`aic`	The AIC of the model.
`aicc`	The AICc of the model.
`method`	The optimization method used to fit the model.
`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.

root_density	A vector giving the density of probability of the trait at the root of the tree. Each element corresponds to one point in the discretized trait grid, from left (lower bound) to right (upper bound).

Author(s)

F. C. Boucher

Examples

```
# Simulate data: tree + continuous trait
library(geiger) # we will use geiger for simulating the tree
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 to a total depth of 100
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5)) # TRAIT simulated on th

# fit a model with a linear  potential: multiple starting points in the optimization to ensure convergence
BBM_x=fit_BBMV(tree,TRAIT,Npts=20,method='Nelder-Mead',verbose=T,V_shape='linear')
```

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
V	A vector giving the evolutionary potential.
bounds	A vector of bounds of the trait interval.

Author(s)

F. C. Boucher

LogLik_bounds *Likelihood calculations for the BBM+V model*

Description

Internal functions that calculate the log-likelihood of various versions of the BBM+V model, used in ML and MCMC estimation.

Usage

```
LogLik_bounds(tree_formatted, dCoeff, dMat, 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.
Npts	The number of points used for discretizing the trait interval.

Author(s)

F. C. Boucher

log_prior_7pars_root_bounds *Prior function.*

Description

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

Usage

```
log_prior_7pars_root_bounds(type = c(rep("Normal", 4), rep("Uniform", 3)), shape = list(c(0, 10),
```

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_int	The number of points on the grid between min(trait) and max(trait).
trait	A named vector of trait values for the tips of the tree.

Author(s)

F. C. Boucher

`MH_MCMC_V_ax4bx2cx_root_bounds` *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_V_ax4bx2cx_root_bounds(tree, trait, Nsteps = 5e+05, record_every = 100, plot_every = 500,
```

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.
`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_int`	The number of points on the grid between min(trait) and max(trait).
`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,Bmin,Bmax).
`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.

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 fist parameter (log(sig^2/2)) is the diffusion coefficient, not the evolutionary rate (sig^2). Finally, be careful that the intial bounds you propose actually 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 for bounds and x0 is discrete uniform priors, 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. Values for the priors on the bounds (forced to be discrete uniform) are a single numeric value giving the maximum number of points that can be added on the trait grid outside of the observed trait interval, with a default value of 10 points.

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, Bmin, and Bmax 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.
`bmin`	The value of the lower bound of the trait interval.
`bmax`	The value of the upper bound of the trait interval.
`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

```
# Simulate data: tree + continuous trait
library(geiger) # we will use geiger for simulating the tree
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 to a total depth of 100
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5)) # TRAIT simulated on th
```

```
# MCMC estimation: only 20,000 generations for a quick test
MCMC= MH_MCMC_V_ax4bx2cx_root_bounds(tree,trait=TRAIT,Nsteps=20000,record_every=100,plot_every=500,Npts_int

# Explore MCMC outputs
library(coda)
apply(MCMC[-c(1:50),2:11],2,effectiveSize) # Effective Sample Size for sampling of parameters, ideally we sho

# plot posterior distributions of parameters
par(mfrow=c(2,5))
hist(log(MCMC[-c(1:50),2]/2),breaks=100,main='log(sigsq/2)',ylab=NULL)
hist(MCMC[-c(1:50),3],breaks=100,main='a (x^4 term)',ylab=NULL)
hist(MCMC[-c(1:50),4],breaks=100,main='b (x^2 term)',ylab=NULL)
hist(MCMC[-c(1:50),5],breaks=100,main='c (x term)',ylab=NULL)
hist(MCMC[-c(1:50),6],breaks=100,main='root',ylab=NULL)
hist(MCMC[-c(1:50),7],breaks=100,main='bmin',ylab=NULL)
hist(MCMC[-c(1:50),8],breaks=100,main='bmax',ylab=NULL)
hist(MCMC[-c(1:50),9],breaks=100,main='lnprior',ylab=NULL)
hist(MCMC[-c(1:50),10],breaks=100,main='lnlik',ylab=NULL)
hist(MCMC[-c(1:50),11],breaks=100,main='quasi-lnpost',ylab=NULL)

# Now we will run a chain with the potential forced to be linear (i.e. what we simulated)
# We do this by fixing the intial values of a and b to 0 and setting their probabilities of update to zero: the
MCMC_trend= MH_MCMC_V_ax4bx2cx_root_bounds(tree,trait=TRAIT,Nsteps=2000,record_every=100,plot_every=500,Npt

# sample size and plots
apply(MCMC_trend[-c(1:50),c(2,5:11)],2,effectiveSize)

par(mfrow=c(2,4))
hist(log(MCMC_trend[-c(1:50),2]/2),breaks=100,main='log(sigsq/2)',ylab=NULL)
hist(MCMC[-c(1:50),5],breaks=100,main='c (x term)',ylab=NULL)
hist(MCMC_trend[-c(1:50),6],breaks=100,main='root',ylab=NULL)
hist(MCMC_trend[-c(1:50),7],breaks=100,main='bmin',ylab=NULL)
hist(MCMC_trend[-c(1:50),8],breaks=100,main='bmax',ylab=NULL)
hist(MCMC[-c(1:50),9],breaks=100,main='lnprior',ylab=NULL)
hist(MCMC_trend[-c(1:50),10],breaks=100,main='lnlik',ylab=NULL)
hist(MCMC_trend[-c(1:50),11],breaks=100,main='quasi-lnpost',ylab=NULL)
```

`Optim_bBM_0_flex_pts_multiple_starts` *Maximum-likelihood estimation*

Description

Set of internal functions that are used for ML optimization of various forms of the BBM+v model.

Usage

```
Optim_bBM_0_flex_pts_multiple_starts(tree, trait, Npts = 50, method = "Nelder-Mead", verbose = T)
```

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.
Npts	The number of points used in the discretization procedure.
method	The optimization routine used.
verbose	If TRUE, prints progress to the screen.
bounds	A vector giving the bounds of the trait interval.
start.point	A vector giving the values of initial parameters from which the optimization procedure starts.

Author(s)

F. C. Boucher

`plot.landscape.BBMV` *Plot adaptive landscapes*

Description

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

Usage

```
plot.landscape.BBMV(model, landscape = T, Npts = 50, main = "Adaptive landscape")
```

Arguments

model	A BBM+V model fit, as returned by fit_BBMV.
models	A list of BBM+V model fits, as returned by fit_BBMV.
landscape	If TRUE (the default), plots the adaptive landscape. If FALSE, plots the evolutionary potential instead.
Npts	The number of points used to discretize the trait interval for plotting.
main	A title for the plot.
ylim	The upper limit of the plotting region when multiple adaptive landscapes are plotted together.

Value

A plot of the adaptive landscape (or alternatively the evolutionary potential) across the trait interval.

Author(s)

F. C. Boucher

Examples

```
# Simulate data: tree + continuous trait
library(geiger) # we will use geiger for simulating the tree
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 to a total depth of 100
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5)) # TRAIT simulated on th

# fit a model with a linear  potential: multiple starting points in the optimization to ensure convergence
BBM_x=fit_BBMV(tree,TRAIT,Npts=20,method='Nelder-Mead',verbose=T,V_shape='linear')

# plot the adaptive landscape
plot.landscape.BBMV(model=BBM_x,Npts=100)

# plot the evolutionary potential
plot.landscape.BBMV(model=BBM_x,Npts=100, landscape=F)
```

`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 of bounds of the trait interval.

Author(s)

F. C. Boucher

`proposal_7pars_root_bounds` *Parameter update*

Description

Internal function that proposes parameter updates used in MCMC estimation of the BBM+V model.

Usage

```
proposal_7pars_root_bounds(type = "Uniform", sensitivity, pars, trait, Npts_int)
```

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.
trait	A named vector of trait values for the tips of the tree. It should match tip labels in the phylogeny.
Npts_int	The number of points on the grid between min(trait) and max(trait).

Author(s)

F. C. Boucher

Sim_BBMV *Simulation of the BBM+V process.*

Description

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

Usage

```
Sim_BBMV(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) # we will use geiger for simulating the tree
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 to a total depth of 100

# Simulate a trait evolving on the tree with a linear trend towards small values (potential increases with hig
TRAIT= Sim_BBMV(tree,x0=0,V=seq(from=0,to=5,length.out=50),sigma=10,bounds=c(-5, 5))
hist(TRAIT)
```

`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.
`V`	The evolutionary potential used
`bounds`	A vector with the values of both bounds.

Author(s)

F. C. Boucher

Index