Package ‘jive’

July 10, 2014

Type Package

Title Analysis of joint intra- and interspecific trait evolution

Version 0.1

Date 2014-06-05

Author Anna Kostikova <anna.kostikova@unil.ch>, Daniele Silvestro <daniele.Silvestro@unil.ch>, Sacha Laurent <Sacha.Laurent@unil.ch>, Martha Serrano <martha.serrano@unil.ch>, Glenn Litsios <glenn.litsios@unil.ch>,Wim Hordijk <wim@WorldWideWanderings.net>,Nicolas Salamin <nicolas.salam

Maintainer Anna Kostikova <anna.kostikova@unil.ch>

Description macroevolutionary analysis of joint intra- and interspecific variance evolution using comparative phylogenetic models. In addition, it can traitgrams through time based on various comparative models such as Brownian motion, Ornstein-Uhlenbeck, etc.

Depends R (>= 3.0.3)

Imports
OUwie,nloptr,ape,MASS,phytools,TeachingDemos,hdrcde,akima,coda,scales,TreeSim,gtools,phytools

License GPL-2

Repository CRAN

R topics documented:

classicalModel *Simulate trees*

Description

Simulate species trees with complex diversification scenarios

Usage

```
classicalModel(nb.divers.shifts, nb.mass.ext, nb.spec, lambda.max, lambda.min,
  mu.min, surv.rate.min, seed = 1)
```

Arguments

`nb.divers.shifts`	number of diversification shifts on the tree
`nb.mass.ext`	number of mass extinction events
`nb.spec`	total number of species in the final tree
`lambda.max`	maximum value of speciation rate
`lambda.min`	minimum value of speciation rate
`mu.min`	minimum value of extinction rate
`surv.rate.min`	survival rate
`seed`	seed value for reproducibility

Details

The aim of the function is to simulate species trees with complex diversification scenarios, with changing diversification rates at inner nodes of the phylogeny and mass extinction events at particular random times in the past. The user can choose the total number of species, the number of diversification rate changes, the number of mass extinction events, the minimum and maximum values for lambda and mu, and the minimum survival rate for mass extinction events. Then the actual rates are choosen randomly. User gets as output the tree simulated, and all the parameters used for its simulation.

Value

An object of class phylo and a set of parameters used for simulation.

Author(s)

Sacha Laurent

Examples

```
tree <- classicalModel(1, 1, 500, 0.1, 0.01, .01, .2, seed=1506621)
```

`jiveMake` *Make jive object*

Description

This function makes a jive object from a matrix of intraspecific observations and species phylogeny. The obtained jive object can than be used as an input to `jiveMCMC` function Intraspecific observations should be stored as matrix, where lines are vector of observations for each species, with NA for no data. Phylogenetic tree can be either a simmap object (`make.simmap`) or phylo object (`as.phylo`)

Usage

```
jiveMake(simmap, traits, model.var = "OU1", model.mean = "BM",
  model.lik = "Multinorm")
```

Arguments

`simmap`	an object of class "jive" (see details)
`traits`	name of the output file that will store the log of MCMC chain
`model.var`	sampling frequency of the MCMC chain (how often chain will be saved into output file
`model.mean`	printing frequency of the MCMC chain (how often chain will be printed in the R console)
`model.lik`	number of classes for thermodynamic integration (see details)

Details

This function creates a jive object needed for `jiveMCMC` function. Trait values must be stored as a matrix, where lines are vectors of observations for each species, with NA for no data. Rownames are species names. Phylogenetic tree must be provided as either simmap object (for models with multiple regimes) or as a phylo object (for BM or OU1 models). Rownames and tip labels of a phylogenetic tree should match exactly. There are three models implemeted for estimation of species variances evolution - BM, OU1 and OUM. Evolution of species means is only implemented with BM model. Species-specific distribution are models as multivariate normal distribution

Value

An object of class jive

Author(s)

Anna Kostikova

Examples

```
## number of species we want to simulate
n <- 50

## generate tree with a pure birth model and scale it to the height of 1
tree <- pbtree(b = 1, n = n, scale = 1, nsim = 1, ape = TRUE)
```

```
## set parameters for OU1 model of species-specific variances
sig.sq <- 0.9
alpha  <- 0.1
theta0 <- 1
theta  <- 5

## set parameters for BM model of specific-specific means
sig.sq.bm <- 0.5
mu0       <- 350

## set mean number of observations per species
mean.obs <- 20

## get selective regimes (all 1s because of OU1 model)
y <- data.frame(tree$tip.label, rep(1, n))

## add node labels
tree$node.label <- rep("1", n-1)

## simulate species-specific variances
sigma.val <- abs(OUwie.sim(tree, y, simmap.tree=FALSE,
scaleHeight=TRUE, alpha=rep(alpha,2),
sigma.sq=rep(sig.sq,2), theta0=theta0, theta=theta)$X)

## simulate species-specific means
mean.val <- mvrnorm(mu=rep(mu0, length(tree$tip)), Sigma=(sig.sq.bm * vcv(tree)))

## draw a random number of intraspecific observations for each species
spec.obs <- rpois(n, mean.obs)

## generate a data matrix where rows are species and columns are individual observations
traits <- matrix(rnorm(M.spec.obs * n, mean=mean.val, sd=sqrt(sigma.val)),
nrow=n, ncol=max(spec.obs))
traits <- cbind(as.matrix(max(spec.obs) - spec.obs), traits)

## function to replace empty cells with NA
foo <- function(x){
to <- x[1]
x[1:(to + 1)] <- NA
return(x[-1])
}

## apply to data matrix
traits <- as.matrix(t(apply(traits, 1, foo)))

## add species names to rownames
rownames(traits) <- tree$tip.label
my.jive <- jiveMake(tree, traits,  model.var="OU1", model.mean="BM", model.lik="Multinorm")
```

`jiveMCMC` *Jive MCMC*

Description

Implements Markov chain Monte Carlo sampling for trait evolutionary models with intraspecific data

Usage

```
jiveMCMC(jive, log.file = "jive_mcmc.log", sampling.freq = 1000,
  print.freq = 1000, ncat = 1, beta.param = 0.3, ngen = 5500000,
  burnin = 0, update.freq = NULL)
```

Arguments

`jive`	an object of class "jive" (see details)
`log.file`	name of the output file that will store the log of MCMC chain
`sampling.freq`	sampling frequency of the MCMC chain (how often chain will be saved into output file
`print.freq`	printing frequency of the MCMC chain (how often chain will be printed in the R console)
`ncat`	number of classes for thermodynamic integration (see details)
`beta.param`	beta value to define classes for thermodynamic integration (see details)
`ngen`	number of generation in MCMC chain
`burning`	a burning phase of MCMC chain (has to be specified for thermodynamic integration)
`update.freq`	update frequencies for likelihood and prior level parameters

Details

This function runs MCMC sampling on jive object `make.jive`. The jive object contains both the dataset and set of model to be used in MCMC. This function implements both a conventional MCMC and an MCMC with thermodynamic integration. The latter option is turned off by default and can be changed by setting ncat to values > 1. The recommended ncat for TI is 10. When setting ncat > 1, make sure to specify brunin. Larger ncat values will result in longer MCMC running time (?).

Examples

```
jiveMCMC(my.jive, log.file="ny.jive_MCMC.log")
```

`jivePlot` *Plot jive MCMC object*

Description

Creates a jive object from intraspecific observations and species phylogeny

Usage

```
jivePlot(tree, proc.jive, regime, cols = c("blue", "green"),
  cex.label = 0.7, cex.circle = 2, lab.off = 0.025, ladder = TRUE, ...)
```

Arguments

`tree`	an object of class "jive" (see details)
`proc.jive`	name of the output file that will store the log of MCMC chain
`regime`	sampling frequency of the MCMC chain (how often chain will be saved into output file
`cols`	printing frequency of the MCMC chain (how often chain will be printed in the R console)
`cex.label`	number of classes for thermodynamic integration (see details)
`cex.circle`	
`lab.off`	
`ladder`	

Details

This function creates a jive object needed for `jiveMCMC` function. Trait values must be stored as a matrix, where lines are vector of observations for each species, with NA for no data. Rownames are species names. Phylogenetic tree must be provided as either simmap object (for models with multiple regimes) or as a phylo object (for BM or OU1 models)

Examples

```
jivePlot(phy1, my.l, regime)
```

`jiveProc` *Process jive MCMC*

Description

Process the jiveMCMC output log file

Usage

```
jiveProc(log.file = "jive_mcmc_OU1.log", n.spec, stat = jiveMode,
  burning = 0, probHPD = 0.95, verbose = TRUE, ...)
```

Arguments

`log.file`	log file recorded by `jiveMCMC` function
`n`	number of species
`stat`	which statistics to use to summarize MCMC. By default, set to mode for prior level parameters and mean for likelihood level parameters. Can also be `mean`, `median`.
`burning`	how much of burning to disregard
`probHPD`	set HPD intervals
`verbose`	how much of statistics to return
`...`	additional parameters that can be passed to HPDinterval function

Details

This function processes the output log file of the jiveMCMC function. It summarizes posterior sample for each variable into summary statistics (e.g. mean, mode, median) and calculates HPD invervals

Value

A list of averaged statistics from MCMC chain for each parameter (list)

Author(s)

Anna Kostikova

Examples

```
my.summary <- jiveProc(log.file="OU_log.log", n = 50, verbose=FALSE)
```

---

make.hpfun *Hyper-prior function*

Description

This function creates a hyper-prior density function. Currently supported density function are Uniform, Gamma and Normal. The resulting function is used during MCMC jiveMCMC to estimate parameters of priors.

Usage

```
make.hpfun(hpf = "Uniform", hp.pars, ...)
```

Arguments

`hpf`	name of a density function. Supported density functions are: Uniform, Gamma and Normal
`hp.pars`	a vector of density function parameters
`...`	additional parameters that can be passed to a density function

Details

There are three currently implemented density function: Uniform, Gamma and Normal. Each of these densities requires two input parameters and hp.pars must be a vector of two values and cannot be left empty.

Value

Hyper-prior density function (function)

A hyper-prior density function (function)

Author(s)

Anna Kostikova and Daniele Silvestro

Examples

```
my.hp <- make.hpfun(hpf="Uniform", hp.pars=c(1,2))
```

`slidingWin` *Proposal functions*

Description

Multiplier and sliding window proposals

Usage

```
slidingWin(i, d)
```

Arguments

`i`	fff
`d`	fff
`...`	ff

Details

Make a function to get a hyper prior

Examples

```
sw <- slidingWin(i=2, d=9)
```

Index