Package 'bayou'

October 25, 2015

Type Package

Title Bayesian Fitting of Ornstein-Uhlenbeck Models to Phylogenies

Version 1.1.0

Date 2015-10-16

Author Josef C. Uyeda, Jon Eastman and Luke Harmon

Maintainer Josef C. Uyeda <josef.uyeda@gmail.com>

Description Tools for fitting and simulating multi-optima Ornstein-Uhlenbeck models to phylogenetic comparative data using Bayesian reversible-jump methods.

License GPL (>= 2)

Depends ape (>= 3.0-6), geiger(>= 2.0), R (>= 2.15.0), phytools, coda

Imports Rcpp (>= 0.10.3), MASS, mnormt, fitdistrplus, denstrip, foreach, stats, grDevices, graphics

Suggests doParallel

LinkingTo Rcpp, RcppArmadillo

Collate 'RcppExports.R' 'bayou-utilities.R' 'probability.R' 'bayou-weight_matrix.R' 'bayou-moves.R' 'bayou-likelihood.R' 'bayou-prior.R' 'conversion-utilities.R' 'bayou-mcmc-utilities.R' 'bayou-mcmc.R' 'bayou-plotting.R' 'bayou-simulation.r' 'bayou-steppingstone.R' 'bayou-package.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-10-25 08:55:52

R topics documented:

bayou-package *bayou-package*

Description

A package for inferring adaptive evolution to phylogenetic comparative data using Bayesian reversible-jump estimation of multi-optima Ornstein-Uhlenbeck models.

Author(s)

Josef C Uyeda

bayou.lik *Function for calculating likelihood of an OU model in bayou using the threepoint algorithm*

Description

Function for calculating likelihood of an OU model in bayou using the threepoint algorithm

Usage

```
bayou.lik(pars, cache, X, model = "OU")
```

Arguments

pars	A list of parameters to calculate the likelihood
cache	A bayou cache object generated using .prepare.ou.univariate
X	A named vector giving the tip data
model	Parameterization of the OU model. Either "OU", "QG" or "OUrepar".

Details

This function implements the algorithm of Ho and Ane (2014) implemented in the package phylolm for the OUfixedRoot model. It is faster than the equivalent pruning algorithm in geiger, and can be used on non- ultrametric trees (unlike OU.lik, which is based on the pruning algorithm in geiger).

bayou.mcmc *Bayesian sampling of multi-optima OU models*

Description

Runs a reversible-jump Markov chain Monte Carlo on continuous phenotypic data on a phylogeny, sampling possible shift locations and shift magnitudes, and shift numbers.

Usage

```
bayou.mcmc(tree, dat, SE = 0, model = "OU", prior, ngen = 10000,
  samp = 10, chunk = 100, control = NULL, tuning = NULL,
  new.dir = FALSE, plot.freq = 500, outname = "bayou",
  ticker.freq = 1000, tuning.int = c(0.1, 0.2, 0.3), startpar = NULL,
  moves = NULL, control.weights = NULL, lik.fn = NULL)
```

Arguments

tree	a phylogenetic tree of class 'phylo'
dat	a named vector of continuous trait values matching the tips in tree
SE	The standard error of the data. Either a single value applied to all the data, or a vector of length(dat).
model	The parameterization of the OU model used. Either "OU" for standard parameterization with alpha and sigma^2; "OUrepar" for phylogenetic half-life and stationary variance (Vy), or "QG" for the Lande model, with parameters h^2 (heritability), P (phenotypic variance), omega^2 (width of adaptive landscape), and Ne (effective population size)
prior	A prior function of class 'priorFn' that gives the prior distribution of all parameters
ngen	The number of generations to run the Markov Chain
samp	The frequency at which Markov samples are retained
chunk	The number of samples retained in memory before being written to a file
control	A list providing a control object governing how often and which proposals are used
tuning	A named vector that governs how liberal or conservative proposals are that equals the number of proposal mechanisms.
new.dir	If TRUE, then results are stored in a new temporary directory. If FALSE, results are written to the current working directory. If a character string, then results are written to that working directory.
plot.freq	How often plots should be made during the mcmc. If NULL, then plots are not produced
outname	The prefix given to files created by the mcmc
ticker.freq	How often a summary log should be printed to the screen

tuning.int	How often the tuning parameters should be adjusted as a fraction of the total number of generations (currently ignored)
startpar	A list with the starting parameters for the mcmc. If NULL, starting parameters are simulated from the prior distribution
moves	A named list providing the proposal functions to be used in the mcmc. Names correspond to the parameters to be modified in the parameter list. See 'details' for default values.
control.weights	A named vector providing the relative frequency each proposal mechanism is to be used during the mcmc
lik.fn	Likelihood function to be evaluated. Defaults to bayou.lik.

Details

By default, the alpha, sig2 (and various reparameterizations of these parameters) are adjusted with multiplier proposals, theta are adjusted with sliding window proposals, and the number of shifts is adjusted by splitting and merging, as well as sliding the shifts both within and between branches. Allowed shift locations are specified by the prior function (see make.prior()).

---

bayou2OUwie *Converts bayou data into OUwie format*

Description

bayou2OUwie Converts a bayou formatted parameter list into OUwie formatted tree and data table that can be analyzed in OUwie

Usage

```
bayou2OUwie(pars, tree, dat)
```

Arguments

pars	A list with parameter values specifying sb = the branches with shifts, loc = the location on branches where a shift occurs and t2 = the optima to which descendants of that shift inherit
tree	A phylogenetic tree
dat	A vector of tip states

Value

A list with an OUwie formatted tree with mapped regimes and an OUwie formatted data table

cdpois *Conditional Poisson distribution*

Description

cdpois calculates the probability density of a value k from a Poisson distribution with a maximum kmax. rdpois draws random numbers from a conditional Poisson distribution.

Usage

```
cdpois(k, lambda, kmax, log = TRUE)

rdpois(n, lambda, kmax, ...)
```

Arguments

k	random variable value
lambda	rate parameter of the Poisson distribution
kmax	maximum value of the conditional Poisson distribution
log	log transformed density
n	number of samples to draw
...	additional parameters passed to dpois or rpois

Examples

```
cdpois(10,1,10)
cdpois(11,1,10)
#rdpois(5,10,10)
```

combine.chains *Combine mcmc chains*

Description

Combine mcmc chains

Usage

```
combine.chains(chain1, chain2, burnin.prop = 0)
```

Arguments

chain1	The first chain to be combined
chain2	The second chain to be combined
burnin.prop	The proportion of burnin from each chain to be discarded

Value

A combined bayouMCMC chain

`dataSim` *Simulates data from bayou models*

Description

This function simulates data for a given set of parameter values.

Usage

```
dataSim(pars, model, tree, map.type = "pars", SE = 0, phenogram = TRUE,
  ...)
```

Arguments

`pars`	A bayou formated parameter list
`model`	The type of model specified by the parameter list (either "OU", "OUrepar" or "QG").
`tree`	A tree of class 'phylo'
`map.type`	Either "pars" if the regimes are taken from the parameter list, or "simmap" if taken from the stored simmap in the tree
`SE`	A single value or vector equal to the number of tips specifying the measurement error that should be simulated at the tips
`phenogram`	A logical indicating whether or not the simulated data should be plotted as a phenogram
`...`	Optional parameters passed to `phenogram(...)`.

Details

`dataSim` Simulates data for a given bayou model and parameter set

dhalfcauchy — *Half cauchy distribution taken from the R package LaplacesDemon (Hall, 2012).*

Description

dhalfcauchy returns the probability density for a half-Cauchy distribution

Usage

```
dhalfcauchy(x, scale = 25, log = FALSE)

phalfcauchy(q, scale = 25)

qhalfcauchy(p, scale = 25)

rhalfcauchy(n, scale = 25)
```

Arguments

x	A parameter value for which the density should be calculated
scale	The scale parameter of the half-Cauchy distributoin
log	A logical indicating whether the log density should be returned
q	A vector of quantiles
p	A vector of probabilities
n	The number of observations

dloc — *Probability density function for the location of the shift along the branch*

Description

Since unequal probabilities are incorporated in calculating the density via dsb, all branches are assumed to be of unit length. Thus, the dloc function simply returns 0 if log=TRUE and 1 if log=FALSE.

Usage

```
dloc(loc, min = 0, max = 1, log = TRUE)

rloc(k, min = 0, max = 1)
```

Arguments

loc	The location of the shift along the branch
min	The minimum position on the branch the shift can take
max	The maximum position on the branch the shift can take
log	A logical indicating whether the log density should be returned
k	The number of shifts to return along a branch

Details

dloc calculates the probability of a shift occuring at a given location along the branch assuming a uniform distribution of unit length rloc randomly generates the location of a shift along the branch

---

dsb *Probability density functions for bayou*

Description

This function provides a means to specify the prior for the location of shifts across the phylogeny. Certain combinations are not allowed. For example, a maximum shift number of Inf on one branch cannot be combined with a maximum shift number of 1 on another. Thus, bmax must be either a vector of 0's and Inf's or a vector of 0's and 1's. Also, if bmax == 1, then all probabilities must be equal, as bayou cannot sample unequal probabilities without replacement.

Usage

```
dsb(sb, ntips = ntips, bmax = 1, prob = 1, log = TRUE)

rsb(k, ntips = ntips, bmax = 1, prob = 1, log = TRUE)
```

Arguments

sb	A vector giving the branch numbers (for a post-ordered tree)
ntips	The number of tips in the phylogeny
bmax	A single integer or a vector of integers equal to the number of branches in the phylogeny indicating the maximum number of shifts allowable in the phylogeny. Can take values 0, 1 and Inf.
prob	A single value or a vector of values equal to the number of branches in the phylogeny indicating the probability that a randomly selected shift will lie on this branch. Can take any positive value, values need not sum to 1 (they will be scaled to sum to 1)
log	A logical indicating whether the log probability should be returned. Default is 'TRUE'
k	The number of shifts to randomly draw from the distribution

Details

dsb calculates the probability of a particular arrangement of shifts for a given set of assumptions.

Value

The log density of the particular number and arrangement of shifts.

Examples

```
n=10
tree <- sim.bdtree(n=n)
tree <- reorder(tree, "postorder")
nbranch <- 2*n-2
sb <- c(1,2, 2, 3)

# Allow any number of shifts on each branch, with
# probability proportional to branch length
dsb(sb, ntips=n, bmax=Inf, prob=tree$edge.length)

# Disallow shifts on the first branch, returns -Inf
# because sb[1] = 1
dsb(sb, ntips=n, bmax=c(0, rep(1, nbranch-1)),
                            prob=tree$edge.length)

# Set maximum number of shifts to 1, returns -Inf
# because two shifts are on branch 2
dsb(sb, ntips=n, bmax=1, prob=1)

#Generate a random set of k branches
rsb(5, ntips=n, bmax=Inf, prob=tree$edge.length)
```

---

gelman.R *Calculate Gelman's R statistic*

Description

Calculate Gelman's R statistic

Usage

```
gelman.R(parameter, chain1, chain2, freq = 20, start = 1, plot = TRUE,
  ...)
```

Arguments

parameter	The name or number of the parameter to calculate the statistic on
chain1	The first bayouMCMC chain
chain2	The second bayouMCMC chain

freq	The interval between which the diagnostic is calculated
start	The first sample to calculate the diagnostic at
plot	A logical indicating whether the results should be plotted
...	Optional arguments passed to gelman.diag(...) from the coda package

---

identifyBranches *Identify shifts on branches of a phylogenetic tree*

Description

This is a convenience function for mapping regimes interactively on the phylogeny. The method locates the nearest branch to where the cursor is clicked on the plot and records the branch number and the location selected on the branch.

Usage

```
identifyBranches(tree, n, fixed.loc = TRUE, plot.simmap = TRUE)
```

Arguments

tree	An object of class 'phylo'
n	The number of shifts to map interactively onto the phylogeny
fixed.loc	A logical indicating whether the exact location on the branch should be returned, or the shift will be free to move along the branch
plot.simmap	A logical indicating whether the resulting painting of regimes should be plotted following the selection shift location.

Details

identifyBranches opens an interactive phylogeny plot that allows the user to specify the location of shifts in a phylogenetic tree.

Value

Returns a list with elements "sb" which contains the branch numbers of all selected branches with length "n". If "fixed.loc=TRUE", then the list also contains a vector "loc" which contains the location of the selected shifts along the branch.

load.bayou *Loads a bayou object*

Description

load.bayou loads a bayouFit object that was created using bayou.mcmc()

Usage

```
load.bayou(bayouFit, save.Rdata = TRUE, file = NULL, cleanup = FALSE)
```

Arguments

bayouFit	An object of class bayouFit produced by the function bayou.mcmc()
save.Rdata	A logical indicating whether the resulting chains should be saved as an *.rds file
file	An optional filename (possibly including path) for the saved *.rds file
cleanup	A logical indicating whether the files produced by bayou.mcmc() should be removed.

Details

If both save.Rdata is FALSE and cleanup is TRUE, then load.bayou will trigger a warning and ask for confirmation. In this case, if the results of load.bayou() are not stored in an object, the results of the MCMC run will be permanently deleted.

Examples

```
## Not run:
data(chelonia)
tree <- chelonia$phy
dat <- chelonia$dat
prior <- make.prior(tree)
fit <- bayou.mcmc(tree, dat, model="OU", prior=prior,
                                  new.dir=TRUE, ngen=5000)
chain <- load.bayou(fit, save.Rdata=FALSE, cleanup=TRUE)
plot(chain)

## End(Not run)
```

Lposterior *Return a posterior of shift locations*

Description

Return a posterior of shift locations

Usage

```
Lposterior(chain, tree, burnin = 0, simpar = NULL, mag = TRUE)
```

Arguments

chain	A bayouMCMC chain
tree	A tree of class 'phylo'
burnin	A value giving the burnin proportion of the chain to be discarded
simpar	An optional bayou formatted parameter list giving the true values (if data were simulated)
mag	A logical indicating whether the average magnitude of the shifts should be returned

Value

A data frame with rows corresponding to postordered branches. pp indicates the posterior probability of the branch containing a shift. magnitude of theta2 gives the average value of the new optima after a shift. naive SE of theta2 gives the standard error of the new optima not accounting for autocorrelation in the MCMC and rel location gives the average relative location of the shift on the branch (between 0 and 1 for each branch).

make.powerposteriorFn *Makes a power posterior function in bayou*

Description

This function generates a power posterior function for estimation of marginal likelihood using the stepping stone method

Usage

```
make.powerposteriorFn(k, Bk, priorFn, refFn)
```

Arguments

k	The step in the sequence being estimated
Bk	The sequence of steps to be taken from the reference function to the posterior
priorFn	The prior function to be used in marginal likelihood estimation
refFn	The reference function generated using make.refFn() from a preexisting mcmc chain

Details

For use in stepping stone estimation of the marginal likelihood using the method of Fan et al. (2011).

Value

A function of class "powerposteriorFn" that returns a list of four values: result (the log density of the power posterior), lik (the log likelihood), prior (the log prior), ref the log reference density.

---

make.prior *Make a prior function for bayou*

Description

This function generates a prior function to be used for bayou according to user specifications.

Usage

```
make.prior(tree, dists = list(), param = list(), fixed = list(),
  plot.prior = TRUE, model = "OU")
```

Arguments

tree	A tree object of class "phylo"
dists	A list providing the function names of the distribution functions describing the prior distributions of parameters (see details). If no distributions are provided for a parameter, default values are given. Note that the names are provided as text strings, not the functions themselves.
param	A list providing the parameter values of the prior distributions (see details).
fixed	A list of parameters that are to be fixed at provided values. These are removed from calculation of the prior value.
plot.prior	A logical indicating whether the prior distributions should be plotted.
model	One of three specifications of the OU parameterization used. Takes values "OU" (alpha & sig2), "QG" (h2, P, w2, Ne), or "OUrepar" (halflife,Vy)

Details

Default distributions and parameter values are given as follows: OU: `list(dists=list("dalpha"="dlnorm","dsig2"="dl`
QG: `list(dists=list("dh2"="dbeta","dP"="dlnorm","dw2"="dlnorm","dNe"="dlnorm", "dk"="cdpois",`
OUrepar: `list(dists=list("dhalflife"="dlnorm","dVy"="dlnorm", "dk"="cdpois","dtheta"="dnorm","`

`dalpha`, `dsig2`, `dh2`, `dP`, `dw2`, `dNe`, `dhalflife`, and `dVy` must be positive continuous distributions and provide the parameters used to calculate alpha and sigma^2 of the OU model. `dtheta` must be continuous and describes the prior distribution of the optima. dk is the prior distribution for the number of shifts. For Poisson and conditional Poisson (cdpois) are provided the parameter `lambda`, which provides the total number of shifts expected on the tree (not the rate per unit branch length). Otherwise, dk can take any positive, discrete distribution. dsb indicates the prior probability of a given set of branches having shifts, and is generally specified by the "dsb" function in the bayou package. See the documentation for dsb for specifying the number of shifts allowed per branch, the probability of a branch having a shift, and specifying constraints on where shifts can occur.`"dloc"` indicates the prior probability of the location of a shift within a single branch. Currently, all locations are given uniform density. All distributions are set to return log-transformed probability densities.

Value

returns a prior function of class "priorFn" that calculates the log prior density for a set of parameter values provided in a list with correctly named values.

Examples

```
## Load data
data(chelonia)
tree <- chelonia$phy
dat <- chelonia$dat

#Create a prior that allows only one shift per branch with equal probability
#across branches
prior <- make.prior(tree, dists=list(dalpha="dlnorm", dsig2="dlnorm",
           dsb="dsb", dk="cdpois", dtheta="dnorm"),
             param=list(dalpha=list(meanlog=-5, sdlog=2),
               dsig2=list(meanlog=-1, sdlog=5), dk=list(lambda=15, kmax=200),
                 dsb=list(bmax=1,prob=1), dtheta=list(mean=mean(dat), sd=2)))

#Evaluate some parameter sets
pars1 <- list(alpha=0.1, sig2=0.1, k=5, ntheta=6, theta=rnorm(6, mean(dat), 2),
                sb=c(32, 53, 110, 350, 439), loc=rep(0.1, 5), t2=2:6)
pars2 <- list(alpha=0.1, sig2=0.1, k=5, ntheta=6, theta=rnorm(6, mean(dat), 2),
                sb=c(43, 43, 432, 20, 448), loc=rep(0.1, 5), t2=2:6)
prior(pars1)
prior(pars2) #-Inf because two shifts on one branch

#Create a prior that allows any number of shifts along each branch with probability proportional
#to branch length
prior <- make.prior(tree, dists=list(dalpha="dlnorm", dsig2="dlnorm",
           dsb="dsb", dk="cdpois", dtheta="dnorm"),
             param=list(dalpha=list(meanlog=-5, sdlog=2),
```

```
                dsig2=list(meanlog=-1, sdlog=5), dk=list(lambda=15, kmax=200),
                  dsb=list(bmax=Inf,prob=tree$edge.length),
                    dtheta=list(mean=mean(dat), sd=2)))
prior(pars1)
prior(pars2)

#Create a prior with fixed regime placement and sigma^2 value
prior <- make.prior(tree, dists=list(dalpha="dlnorm", dsig2="fixed",
           dsb="fixed", dk="fixed", dtheta="dnorm", dloc="dunif"),
             param=list(dalpha=list(meanlog=-5, sdlog=2),
               dtheta=list(mean=mean(dat), sd=2)),
                 fixed=list(sig2=1, k=3, ntheta=4, sb=c(447, 396, 29)))

pars3 <- list(alpha=0.01, theta=rnorm(4, mean(dat), 2), loc=rep(0.1, 4))
prior(pars3)

##Return a list of functions used to calculate prior
attributes(prior)$functions

##Return parameter values used in prior distribution
attributes(prior)$parameters
```

`make.refFn` *Make a reference function in bayou*

Description

This function generates a reference function from a mcmc chain for use in marginal likelihood estimation.

Usage

```
make.refFn(chain, prior, burnin = 0.3, plot = TRUE)
```

Arguments

`chain`	An mcmc chain produced by `bayou.mcmc()` and loaded with `load.bayou()`
`prior`	The prior function used to generate the mcmc chain
`burnin`	The proportion of the mcmc chain to be discarded when generating the reference function
`plot`	Logical indicating whether or not a plot should be created

Details

Distributions are fit to each mcmc chain and the best-fitting distribution is chosen as the reference distribution for that parameter using the method of Fan et al. (2011). For positive continuous parameters `alpha, sigma^2, halflife, Vy, w2, Ne`, Log-normal, exponential, gamma and weibull distributions are fit. For continuous distributions `theta`, Normal, Cauchy and Logistic distributions are fit. For discrete distributions, `k`, negative binomial, poisson and geometric distributions are fit. Best-fitting distributions are determined by AIC.

Value

Returns a reference function of class "refFn" that takes a parameter list and returns the log density given the reference distribution. If `plot=TRUE`, a plot is produced showing the density of variable parameters and the fitted distribution from the reference function (in red).

`makeTransparent` *Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)*

Description

Make a color transparent (Taken from an answer on StackOverflow by Nick Sabbe)

Usage

```
makeTransparent(someColor, alpha = 100)
```

Arguments

`someColor`	A color, either a number, string or hexidecimal code
`alpha`	The alpha transparency. The maxColorValue is set to 255.

`OU.lik` *Function for calculating likelihood of an OU model in bayou using pruning algorithm or matrix inversion*

Description

Function for calculating likelihood of an OU model in bayou using pruning algorithm or matrix inversion

Usage

```
OU.lik(pars, tree, X, SE = 0, model = "OU", invert = FALSE)
```

Arguments

`pars`	A list of parameters to calculate the likelihood
`tree`	A phylogenetic tree of class 'phylo'
`X`	A named vector giving the tip data
`SE`	A named vector or single number giving the standard errors of the data
`model`	Parameterization of the OU model. Either "OU", "QG" or "OUrepar".
`invert`	A logical indicating whether the likelihood should be solved by matrix inversion, rather than the pruning algorithm. This is primarily present to test that calculation of the likelihood is correct.

Details

This function can be used for calculating single likelihoods using previously implemented methods. It is likely to become deprecated and replaced by `bayou.lik` in the future, which is based on `phylolm`'s threepoint algorithm, which works on non-ultrametric trees and is substantially faster.

Value

A list returning the log likelihood ("loglik"), the weight matrix ("W"), the optima ("theta"), the residuals ("resid") and the expected values ("Exp").

`OU.repar` *Calculates the alpha and sigma^2 from a parameter list with supplied phylogenetic half-life and stationary variance*

Description

Calculates the alpha and sigma^2 from a parameter list with supplied phylogenetic half-life and stationary variance

Usage

```
OU.repar(pars)
```

Arguments

`pars`	A bayou formatted parameter list with parameters halflife (phylogenetic halflife) and Vy (stationary variance)

Value

A list with values for alpha and sig2.

`OUphenogram` *Experimental phenogram plotting function for set of model of model parameters*

Description

Experimental phenogram plotting function for set of model of model parameters

Usage

```
OUphenogram(pars, tree, dat, SE = 0, regime.col = NULL, ...)
```

Arguments

pars	A bayou formatted parameter list
tree	A tree of class 'phylo'
dat	A named vector of tip data
SE	Standard error of the tip states
regime.col	A named vector of colors equal in length to the number of regimes
...	Optional arguments passed to phenogram()

Details

This is an experimental plotting utility that can plot a phenogram with a given regime painting from a parameter list. Note that it uses optimization of internal node states using matrix inversion, which is very slow for large trees. However, what is returned is the maximum likelihood estimate of the internal node states given the model, data and the parameter values.

Examples

```
## Not run:
tree <- sim.bdtree(n=50)
tree$edge.length <- tree$edge.length/max(branching.times(tree))
prior <- make.prior(tree, dists=list(dk="cdpois", dsig2="dnorm",
           dtheta="dnorm"), param=list(dk=list(lambda=5, kmax=10),
             dsig2=list(mean=1, sd=0.01), dtheta=list(mean=0, sd=3)),
               plot.prior=FALSE)
pars <- priorSim(prior, tree, plot=FALSE, nsim=1)$pars[[1]]
pars$alpha <- 4
dat <- dataSim(pars, model="OU", phenogram=FALSE, tree)$dat
OUphenogram(pars, tree, dat, ftype="off")

## End(Not run)
```

OUwie2bayou *Converts OUwie data into bayou format*

Description

OUwie2bayou Converts OUwie formatted data into a bayou formatted parameter list

Usage

```
OUwie2bayou(tree, trait)
```

Arguments

tree	A phylogenetic tree with states at internal nodes as node labels
trait	A data frame in OUwie format

Value

A bayou formatted parameter list

`overparameterize.startingPoint` *Generate an overparameterized starting point for the MCMC*

Description

This function takes a prior function and generates a starting point that can be entered for `startpar` in the function `bayou.mcmc`

Usage

```
overparameterize.startingPoint(prior, tree, dat)
```

Arguments

`prior`	A prior function
`tree`	A phylogenetic tree of class 'phylo'
`dat`	A named data vector

Details

This function creates an "overparameterized" starting point for running the mcmc. It gives n-1 tips a unique optimum close to the actual data value. This is useful if you expect steep likelihood peaks that may be hard to find, as these often will be easier to access from this overparameterized model. Generally, the overparameterization will have a very high likelihood and a very low prior.

`parmap.W` *Calculate the weight matrix of a set of regimes on a phylogeny*

Description

These functions calculate weight matrices from regimes specified by a bayou formatted parameter list `parmap.W` calculates the weight matrix for a set of regimes from a phylogeny with a stored regime history. `.parmap.W` calculates the same matrix, but without checks and is generally run internally.

Usage

```
parmap.W(tree, pars)
```

Arguments

tree	either a tree of class "phylo" or a cache object produced by bayOU's internal functions. Must include list element 'maps' which is a simmap reconstruction of regime history.
pars	a list of the parameters used to calculate the weight matrix. Only pars$alpha is necessary to calculate the matrix, but others can be present.

Details

.parmap.W is more computationally efficient within a mcmc and is used internally.

---

pars2simmap *Convert a bayou parameter list into a simmap formatted phylogeny*

Description

This function converts a bayou formatted parameter list specifying regime locations into a simmap formatted tree that can be plotted using plotSimmap from phytools or the plotRegimes function from bayou.

Usage

```
pars2simmap(pars, tree)
```

Arguments

pars	A list that contains sb (a vector of branches with shifts), loc (a vector of shift locations), t2 (a vector of theta indices indicating which theta is present after the shift).
tree	A tree of class 'phylo'

Details

pars2simmap takes a list of parameters and converts it to simmap format

Value

A list with elements: tree A simmap formatted tree, pars bayou formatted parameter list, and cols A named vector of colors.

Examples

```
tree <- reorder(sim.bdtree(n=100), "postorder")

pars <- list(k=5, sb=c(195, 196, 184, 138, 153), loc=rep(0, 5), t2=2:6)
tr <- pars2simmap(pars, tree)
plotRegimes(tr$tree, col=tr$col)
```

phenogram.density *Plot a pheongram with the posterior density for optima values*

Description

Plots a phenogram and the posterior density for optima values

Usage

```
phenogram.density(tree, dat, burnin = 0, chain, colors = NULL,
  pp.cutoff = NULL, K = NULL, ...)
```

Arguments

tree	A phylogeny of class 'phylo'
dat	A named vector of tip data
burnin	The initial proportion of the MCMC to be discarded
chain	A bayouMCMC object that contains the results of an MCMC chain
colors	An optional named vector of colors to assign to regimes, NULL results in no regimes being plotted.
pp.cutoff	The posterior probability cutoff value. Branches with posterior probabilities of having a shift above this value will have the average location of the regime shift painted onto the branches.
K	A list with the values of K to be plotted. If NULL all values of K are combined and a total posterior produced. This allows separate lines to be plotted for different numbers of shifts so that the location of optima can be compared, for example, between all samples that have 1 vs. 2 shifts in the posterior.
...	Additional parameters passed to phenogram(...)

plot.bayouMCMC *S3 method for plotting bayouMCMC objects*

Description

S3 method for plotting bayouMCMC objects

Usage

```
## S3 method for class 'bayouMCMC'
plot(x, ...)
```

Arguments

x	A mcmc chain of class 'bayouMCMC' produced by the function bayou.mcmc and loaded into the environment using load.bayou
...	Additional arguments passed to plot.mcmc from the coda package

plot.ssMCMC *S3 method for plotting ssMCMC objects*

Description

S3 method for plotting ssMCMC objects

Usage

```
## S3 method for class 'ssMCMC'
plot(x, ...)
```

Arguments

x	An 'ssMCMC' object
...	Additional arguments passed to plot

Details

Produces 4 plots. The first 3 plot the prior, reference function and likelihood. Different colors indicate different power posteriors for each. These chains should appear to be well mixed. The final plot shows the sum of the marginal likelihood across each of the steps in the stepping stone algorithm.

plotBayoupars *Plot parameter list as a simmap tree*

Description

Plot parameter list as a simmap tree

Usage

```
plotBayoupars(pars, tree, ...)
```

Arguments

pars	A bayou formatted parameter list
tree	A tree of class 'phylo'
...	Additional arguments passed to plotRegimes

plotRegimes *Function to plot the regimes from a simmap tree*

Description

Function to plot the regimes from a simmap tree

Usage

```
plotRegimes(tree, col = NULL, lwd = 1, pal = rainbow, ...)
```

Arguments

tree	A simmap tree of class phylo or simmap with a tree$maps list
col	A named vector of colors to assign to character states, if NULL, then colors are generated from pal
lwd	A numeric value indicating the width of the edges
pal	A color palette function to generate colors if col=NULL
...	Optional arguments that are passed to plot.phylo

Details

This function uses plot.phylo to generate coordinates and plot the tree, but plots the 'maps' element of phytools' simmap format. This provides much of the functionality of plot.phylo from the ape package. Currently, only types 'phylogram', 'unrooted', 'radial', and 'cladogram' are allowed. Phylogenies must have branch lengths.

plotSimmap.mcmc *Plot a phylogenetic tree with posterior probabilities from a bayouMCMC chain (function adapted from phytools' plotSimmap)*

Description

Plot a phylogenetic tree with posterior probabilities from a bayouMCMC chain (function adapted from phytools' plotSimmap)

Usage

```
plotSimmap.mcmc(chain, burnin = NULL, lwd = 1, edge.type = c("theta",
  "none", "regimes", "pp"), pal = rainbow, pp.cutoff = 0.3,
  circles = TRUE, circle.cex.max = 3, circle.col = "red",
  circle.pch = 21, circle.lwd = 0.75, circle.alpha = 100,
  pp.labels = FALSE, pp.col = 1, pp.alpha = 255, pp.cex = 0.75,
  edge.color = 1, parameter.sample = 1000, ...)
```

Arguments

chain	A bayouMCMC chain
burnin	The proportion of runs to be discarded, if NULL, then the value stored in the bayouMCMC chain's attributes is used
lwd	The width of the edges
edge.type	Either "theta" (branches will be colored according to their median value of theta), "regimes" (clades will be assigned to distinct regimes if the posterior probability of a shift on that branch is > pp.cutoff), or "pp" (branches will be colored according to the probability of a shift on that branch). If "none" then edge.color will be assigned to all branches.
pal	A color palette function used to paint the branches (unless edge.type="none")
pp.cutoff	If edge.type=="regimes", the posterior probability above which a shift should be reconstructed on the tree.
circles	a logical value indicating whether or not a circle should be plotted at the base of the node with values that correspond to the posterior probability of having a shift.
circle.cex.max	The cex value of a circle with a posterior probability of 1
circle.col	The color used to fill the circles
circle.pch	the type of symbol used to plot at the node to indicate posterior probability
circle.lwd	the line width of the points plotted at the nodes
circle.alpha	a value between 0 and 255 that indicates the transparency of the circles (255 is completely opaque).
pp.labels	a logical indicating whether the posterior probability for each branch should be printed above the branch
pp.col	The color used for the posterior probability labels
pp.alpha	a logical or numeric value indicating transparency of posterior probability labels. If TRUE, then transparency is ramped from invisible (pp=0), to black (pp=1). If numeric, all labels are given the same transparency. If NULL, then no transparency is given.
pp.cex	the size of the posterior probability labels
edge.color	The color of edges if edge.type="none"
parameter.sample	When edge.type=="theta", the number of samples used to estimate the median "theta" value from each branch. Since this is computationally intensive, this enables you to downsample the chain.
...	Additional arguments passed to ape's plot.phylo

print.bayouFit *S3 method for printing bayouFit objects*

Description

S3 method for printing bayouFit objects

Usage

```
## S3 method for class 'bayouFit'
print(x, ...)
```

Arguments

x	A 'bayouFit' object produced by bayou.mcmc
...	Additional parameters passed to print

print.priorFn *S3 method for printing priorFn objects*

Description

S3 method for printing priorFn objects

Usage

```
## S3 method for class 'priorFn'
print(x, ...)
```

Arguments

x	A function of class 'priorFn' produced by make.prior
...	Additional arguments passed to print

print.refFn *S3 method for printing refFn objects*

Description

S3 method for printing refFn objects

Usage

```
## S3 method for class 'refFn'
print(x, ...)
```

Arguments

x	A function of class 'refFn' produced by make.refFn
...	Additional arguments passed to print

print.ssMCMC *S3 method for printing ssMCMC objects*

Description

S3 method for printing ssMCMC objects

Usage

```
## S3 method for class 'ssMCMC'
print(x, ...)
```

Arguments

x	An ssMCMC object
...	Optional arguments passed to print

priorSim *Simulates parameters from bayou models*

Description

priorSim Simulates parameters from the prior distribution specified by make.prior

Usage

```
priorSim(prior, tree, plot = TRUE, nsim = 1, ...)
```

Arguments

prior	A prior function created by bayou::make.prior
tree	A tree of class 'phylo'
plot	A logical indicating whether the simulated parameters should be plotted
nsim	The number of parameter sets to be simulated
...	Parameters passed on to plotSimmap(...)

Value

A list of bayou parameter lists

pull.pars *Utility function for retrieving parameters from an MCMC chain*

Description

Utility function for retrieving parameters from an MCMC chain

Usage

```
pull.pars(i, chain, model = "OU")
```

Arguments

i	An integer giving the sample to retrieve
chain	A bayouMCMC chain
model	The parameterization used, either "OU", "QG" or "OUrepar"

Value

A bayou formatted parameter list

Examples

```
## Not run:
tree <- sim.bdtree(n=30)
tree$edge.length <- tree$edge.length/max(branching.times(tree))
prior <- make.prior(tree, dists=list(dk="cdpois", dsig2="dnorm",
             dtheta="dnorm"),
               param=list(dk=list(lambda=15, kmax=32),
                 dsig2=list(mean=1, sd=0.01),
                   dtheta=list(mean=0, sd=3)),
                     plot.prior=FALSE)
pars <- priorSim(prior, tree, plot=FALSE, nsim=1)$pars[[1]]
dat <- dataSim(pars, model="OU", phenogram=FALSE, tree)$dat
fit <- bayou.mcmc(tree, dat, model="OU", prior=prior,
             new.dir=TRUE, ngen=5000, plot.freq=NULL)
chain <- load.bayou(fit, save.Rdata=TRUE, cleanup=TRUE)
plotBayoupars(pull.pars(300, chain), tree)

## End(Not run)
```

QG.alpha *Calculates the alpha parameter from a QG model*

Description

Calculates the alpha parameter from a QG model

Usage

```
QG.alpha(pars)
```

Arguments

pars	A bayou formatted parameter list with parameters h2 (heritability), P (phenotypic variance) and w2 (width of adaptive landscape)

Value

An alpha value according to the equation `alpha = h2*P/(P+w2+P)`.

QG.sig2 *Calculates the sigma^2 parameter from a QG model*

Description

Calculates the sigma^2 parameter from a QG model

Usage

```
QG.sig2(pars)
```

Arguments

pars	A bayou formatted parameter list with parameters h2 (heritability), P (phenotypic variance) and Ne (Effective population size)

Value

An sig2 value according to the equation `alpha = h2*P/(Ne)`.

regime.plot *Adds visualization of regimes to a plot*

Description

Adds visualization of regimes to a plot

Usage

```
regime.plot(pars, tree, cols, type = "rect", transparency = 100)
```

Arguments

pars	A bayou formatted parameter list
tree	A tree of class 'phylo'
cols	A vector of colors to give to regimes, in the same order as pars$sb
type	Either "rect", "density" or "lines". "rect" plots a rectangle for the 95% CI for the stationary distribution of a regime. "density" varies the transparency of the rectangles according to the probability density from the stationary distribution. "lines" plots lines for the mean and 95% CI's without filling them.
transparency	The alpha transparency value for the maximum density, max value is 255.

set.burnin — *Set the burnin proportion for bayouMCMC objects*

Description

Set the burnin proportion for bayouMCMC objects

Usage

```
set.burnin(chain, burnin = 0.3)
```

Arguments

chain	A bayouMCMC chain or an ssMCMC chain
burnin	The burnin proportion of samples to be discarded from downstream analyses.

Value

A bayouMCMC chain or ssMCMC chain with burnin proportion stored in the attributes.

simmap.W — *Calculate the weight matrix of a set of regimes on a phylogeny*

Description

These functions calculate weight matrices from regimes specified in phytools' simmap format. `simmap.W` calculates the weight matrix for a set of regimes from a phylogeny with a stored regime history. `.simmap.W` calculates the same matrix, but without checks and is generally run internally.

Usage

```
simmap.W(tree, pars)
```

Arguments

tree	either a tree of class "phylo" or a cache object produced by bayOU's internal functions. Must include list element 'maps' which is a simmap reconstruction of regime history.
pars	a list of the parameters used to calculate the weight matrix. Only pars$alpha is necessary to calculate the matrix, but others can be present.

Details

`.simmap.W` is more computationally efficient within a mcmc and is used internally. The value of `TotExp` is supplied to speed computation and reduce redundancy, and cache objects must be supplied as the phylogeny, and the parameter `ntheta` must be present in the list `pars`.

`steppingstone` *Stepping stone estimation of the marginal likelihood for a bayou model*

Description

Estimates the marginal likelihood of a bayou model by generating mcmc chains for power posteriors for a series of steps from 0 to 1, progressing from a reference distribution to the posterior distribution.

Usage

```
steppingstone(Bk, chain, tree, dat, SE = 0, prior, startpar = NULL,
  burnin = 0.3, ngen = 10000, powerposteriorFn = NULL, parallel = FALSE,
  ...)
```

Arguments

`Bk`	A vector sequence from 0 to 1 that gives the exponents of the power posterior distribution to take from the reference distribution to the reference distribution. (See details)
`chain`	A mcmc chain used to generate the reference distribution (see `make.refFn` for details).
`tree`	A phylogenetic tree of class "phylo"
`dat`	A named vector of continuous trait data
`SE`	A vector giving the standard error of the trait data. If a single number is given, standard errors are assumed to be constant across the phylogeny.
`prior`	The prior function used to generate the mcmc chain.
`startpar`	The starting parameter values to be used. If any parameters are set as "fixed", this should be specified. If `NULL`, then the parameters are drawn from the prior distribution.
`burnin`	The initial proportion of the provided mcmc chain to be discarded when generating the reference function
`ngen`	The number of mcmc generations to be run for each step of the stepping stone alogrithm.
`powerposteriorFn`	The power posterior function to be used. If `NULL`, this is generated from the provided mcmc chain.
`parallel`	A logical indicating whether or not the chains should be run in parallel.
`...`	Other parameters passed to the mcmc algorithm, see `bayou.mcmc()`.

Details

This function estimates the marginal likelihood of a bayou model by using stepping stone estimation from a reference distribution to the posterior distribution using the method of Fan et al. (2011). The vector `Bk` provides a sequence from 0 to 1. The length of this sequence determines the number of mcmc chains that will be run, and the values are used as the exponents of the power posterior function, stepping from purely the reference distribution (k=0) to purely the posterior distribution (k=1). These chains can be run in parallel if `parallel` is set to `TRUE`. The number of cores available is determined by a call to `detectCores`, and can be set by .... Note that when run in parallel, progress within each of the individual mcmc chains will not be reported, and if `ngen` is high, it may take a considerable amount of time to run. Furthermore, if many samples are saved from each mcmc run, and a number of steps along `Bk` is large, the returned object may require a substantial amount of memory.

Value

A list of class "ssMCMC" that provides the log marginal likelihood `lnr`, a list of the individual normalizing constants estimated at each step `lnrk`, a list of the mcmc chains used for importance sampling to estimating the marginal likelihood at each step `chains`, and mcmc fit data from each of the runs `fits`. Note that this object may become quite large if a number of chains are run for many generations. To reduce the number of samples taken, increase the parameter `samp` (default = 10) which sets the frequency at which samples are saved in the mcmc chain.

`summary.bayouMCMC` *S3 method for summarizing bayouMCMC objects*

Description

S3 method for summarizing bayouMCMC objects

Usage

```
## S3 method for class 'bayouMCMC'
summary(object, ...)
```

Arguments

`object`	A bayouMCMC object
`...`	Additional arguments passed to `print`

Value

An invisible list with two elements: `statistics` which provides summary statistics for a bayouMCMC chain, and `branch.posteriors` which summarizes branch specific data from a bayouMCMC chain.

Index