Package ‘auteur’

February 28, 2011

Type Package

Title reversible-jump Bayesian sampler of evolutionary rates

Version 0.11.0223

Date 2011-02-23

Author LJ Harmon, ME Alfaro, AL Hipp, P Joyce, and JM Eastman

Maintainer Jonathan Eastman <jonathan.eastman@gmail.com>

Depends ape, geiger, colorspace, Rcpp

LinkingTo Rcpp

Description Identifies shifts in process of continuous-trait evolution on phylogenetic trees

License GPL (>=2)

LazyLoad yes

Archs i386, x86_64

R topics documented:

auteur — *main: Bayesian Inference of Trait Evolutionary Process*

Description

auteur (**A**ccommodating **U**ncertainty in **T**rait **E**volution **U**sing **R**) performs reversible-jump Markov chain Monte Carlo sampling to identify phylogenetically localized shifts in the process of continuous trait evolution.

Details

Package:	auteur
Type:	Package
Version:	0.11
License:	GPL
LazyLoad:	yes

Author(s)

Luke J Harmon, Andrew L Hipp, Michael E Alfaro, Paul Joyce, and Jonathan M Eastman <jonathan.eastman@gmail.com

auteur-internal — *internal auteur functions*

Description

`auteur-internal` functions are either not typically called by the user or are currently undocumented

Details

This is an internal **auteur** function, not intended to be called directly by the user.

Note

The likelihood expression used in `bm.lik.fn` owes to Felsenstein (1973), and see also O'meara et al. (2006). That used within `ou.lik.fn` owes to Butler and King (2004).

References

- BUTLER MA and AA KING. 2004. Phylogenetic comparative analysis: A modeling approach for adaptive evolution. American Naturalist 164:683-695.
- FELSENSTEIN J. 1973. Maximum likelihood estimation of evolutionary trees from continuous characters. American Journal of Human Genetics 25:471-492.
- O'MEARA BC, C ANE, MJ SANDERSON, and PC WAINWRIGHT. 2006. Testing for different rates of continuous trait evolution using likelihood. Evolution 60:922-933.

`calibrate.jumpsize` *initialize proposal width*

Description

Generates a reasonable proposal width to initiate sampling for Markov sampling

Usage

```
calibrate.jumpsize(phy, dat, nsteps = 100, model, jumpsizes = NULL)
```

Arguments

`phy`	a phylogenetic tree of class 'phylo'
`dat`	a named vector of continuous trait values, associated with each species in `phy`
`nsteps`	number of proposal steps over which to assess proposal widths
`model`	currently either `model="BM"` or `model="OU"` for Brownian motion or an Ornstein-Uhlenbeck process; see `rjmcmc.bm` and `rjmcmc.ou` for further information on fitting these models
`jumpsizes`	if unspecified, a series of jumpsizes from 1/8 to 8 will be considered

Details

This is primarily an internal function, although may be useful for constraining subsequent runs after an adequate proposal width has been approximated by this function. This function is called internally by `rjmcmc`, which causes additional sampling (one-tenth total Markov chain length) to be prepended to a run. The sole purpose of this initial sampling period is to optimize the proposal width (`jumpsize`) to ensure that the Markov chain achieves convergence. Estimates from this calibration are not stored and do not become available to the user.

Author(s)

JM Eastman

Examples

```
n=4
phy=rescaleTree(phy=rcoal(n=n),totalDepth=100)
dat=rTraitCont(phy=phy, model="BM", sigma=sqrt(0.1))
r=paste(sample(letters,9,replace=TRUE),collapse="")

## with calibrated jumpsize
rjmcmc.bm(phy=phy, dat=dat, ngen=5000, sample.freq=10, prob.mergesplit=0.05, jumpsize=NUI

## with enforced (and large) jumpsize
rjmcmc.bm(phy=phy, dat=dat, ngen=5000, sample.freq=10, prob.mergesplit=0.05, jumpsize=10,

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`compare.rates` *statistical comparison of posterior rate estimates*

Description

Conducts randomization test to determine if posterior rate estimates for two groups of phylogenetic branches differ

Usage

```
compare.rates(branches = list(A = c(NULL, NULL), B = c(NULL, NULL)), phy, poster
```

Arguments

`branches`	a list of numeric labels for branches in the phylogeny, consistent with the node labels found in `phy$edge` (see `read.tree`)
`phy`	a phylogenetic tree of class 'phylo'
`posterior.values`	a dataframe of posterior rate estimates, where rows are sampled generations in the Markov chain and columns distinguish branches in the phylogeny
`...`	additional arguments to be passed to `randomization.test`

Details

For each generation in the Markov sample, a scaled relative rate is computed for each collection of branches (so defined by the supplied list of `branches`). Scaling occurs by weighting posterior rate estimates by the length of the each branch in the group. In so doing, longer branches contribute greater weight to the scaled measure of evolutionary rate in each group. Values need not be rates, but elements must be capable of being associated with each branch in the phylogeny.

Value

A list of scaled relative values for each branch and the result of statistical comparison:

`scl.A`	scaled posterior rate estimates from each Markov sample for group 'A'
`scl.B`	scaled posterior rate estimates from each Markov sample for group 'B'
`r.test`	probability value of randomization comparison between groups for phylogenetically-scaled values, as default under a two-tailed test

Author(s)

JM Eastman

Examples

```
## generate tree
n=24
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))

## simulate a rate shift
 # find an internal edge
```

```
anc=get.desc.of.node(Ntip(phy)+1,phy)
branches=phy$edge[,2]
branches=branches[branches>Ntip(phy) & branches!=anc]
branch=branches[sample(1:length(branches),1)]
desc=get.descendants.of.node(branch,phy)
rphy=phy
rphy$edge.length[match(desc,phy$edge[,2])]=phy$edge.length[match(desc,phy$edge[,2])]*64
e=numeric(nrow(phy$edge))
e[match(c(branch,desc),phy$edge[,2])]=1
cols=rev(diverge_hcl(n=2))
dev.new()
plot(phy,edge.col=ifelse(e==1,cols[1],cols[2]), edge.width=2)
mtext("expected pattern of rates")

## simulate data on the 'rate-shifted' tree
dat=rTraitCont(phy=rphy, model="BM", sigma=sqrt(0.1))

## run reversible-jump MCMC for a short chain
r=paste(sample(letters,9,replace=TRUE),collapse="")
rjmcmc.bm(phy=phy, dat=dat, ngen=10000, sample.freq=10, prob.mergesplit=0.1, fileBase=r)

 # collect posterior rates
load(paste(paste("BM",r,"parameters",sep="."),paste(r,"posteriorsamples.rda",sep="."),sep
rates=posteriorsamples$rates
burnin=round(0.5*nrow(rates))
rates=rates[-c(1:burnin),]

# compare the shifted branches to the background
shift=desc
branches=1:nrow(phy$edge)
background=branches[is.na(match(branches,shift))]

# exclude stem branch from background
background=background[-(which(background==(n+1)))]
comp=compare.rates(branches=list(A=shift, B=background), phy=phy, posterior.values=rates)

# plot posterior distributions of scaled rates by group
prates=data.frame(comp$scl.A, comp$scl.B)
names(prates)=c("shift","background")
trace.plot(obj=data.frame(comp$scl.A, comp$scl.B), col=c("red","black"), alpha=0.4, log="

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
# unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`intercalate.samples` *pool data*

Description

Combines datasets respecting indices of each

Usage

```
intercalate.samples(list.obj)
```

Arguments

`list.obj` a list of vectors or dataframes

Value

A dataframe of pooled values, whose elements are ordered as given by the order of objects in `list.obj`

Author(s)

JM Eastman

Examples

```
# with character vectors
a=letters[1:10]
A=toupper(letters[1:10])
intercalate.samples(list(a,A))

# with multi-column dataframes
list.obj=lapply(1:5, function(x) {y=data.frame(matrix(data=runif(9)+x, nrow=3)); names(y)
intercalate.samples(list.obj)
```

`pool.rjmcmcsamples` *combining posterior samples from Bayesian analysis*

Description

Generates a pooled sample of posterior estimates from any number of independent runs

Usage

```
pool.rjmcmcsamples(base.dirs, lab = "")
```

Arguments

`base.dirs` a vector of directory names, in which the results from `rjmcmc` are to be found

`lab` an optional string used to name the directory to which pooled results are written

Examples

```
n=24
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))
dat=rTraitCont(phy=phy, model="BM", sigma=sqrt(0.1))

## run three short reversible-jump Markov chains
r=paste(sample(letters,9,replace=TRUE),collapse="")
lapply(1:3, function(x) rjmcmc.bm(phy=phy, dat=dat, ngen=500, sample.freq=10, prob.merges

 # collect directories
dirs=sapply(1:3, function(x) dir("./",pattern=paste("BM",r,x,sep=".")))
pool.rjmcmcsamples(base.dirs=dirs, lab=r)
```

```
 # plot density of sampled mean rates across tree
rates=read.table(file=paste(dir(pattern=paste(r,"combined.rjmcmc",sep=".")),paste(r,"rjmc
trace.plot(rates,col="maroon",alpha=0.4,xlab="rates",log="x",legend.control=list(plot=FAL

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`randomization.test` *statistical comparison of sets of values by randomization*

Description

Compares means by bootstrap resampling of differences between empirical distributions

Usage

```
randomization.test(obs = obs, exp = exp, mu = 0, iter = 10000, two.tailed = FALS
```

Arguments

`obs`	a vector of numeric values
`exp`	a vector of numeric values
`mu`	the true difference in means
`iter`	number of randomization comparisons to perform
`two.tailed`	as default, the test is performed under a one-tailed assumption; if `two.tailed=FALSE`, probability values associated with either tail of the comparison distribution are returned, otherwise, a two-tailed result is returned

Details

If a single value is supplied for `obs`, this test equates to finding the quantile in `exp` in which `obs` would be found (under a one-tailed test); see **Examples** and also `ecdf`

Value

A list, whose contents are determined by the above argument:

`unnamed value`	if `two.tailed=TRUE`, this is the two-tailed p-value
`diffs`	the full resampling distribution of differences between `obs` and `exp`, given `mu`
`greater`	if `two.tailed=FALSE`, this is the p-value associated with the righthand tail
`lesser`	if `two.tailed=FALSE`, this is the p-value associated with the lefthand tail

Author(s)

JM Eastman

Examples

```
# a comparison between two distributions
a=rnorm(n=1000, mean=1, sd=0.5)
b=rnorm(n=1000, mean=0, sd=1)
randomization.test(obs=a, exp=b, two.tailed=FALSE)

# a comparison of a single value to a normal distribution
a=3
b=rnorm(n=1000, mean=0, sd=1)
randomization.test(obs=a, exp=b, two.tailed=FALSE)

# compare above result with ecdf(), in which we compute an empirical
f=ecdf(b)
print(1-f(a)) # analogous to a one-tailed test as above
```

`rjmcmc.bm` *Bayesian sampling of shifts in trait-evolutionary rates: Brownian motion*

Description

Implements reversible-jump Markov chain Monte Carlo sampling to identify shifts in the process of continuous trait evolution across phylogenetic trees

Usage

```
rjmcmc.bm(phy, dat, SE=0, ngen = 1000, sample.freq = 100, prob.mergesplit = 0.05
```

Arguments

`phy`	a phylogenetic tree of class 'phylo'
`dat`	a named vector of continuous trait values, associated with each species in `phy`; see `name.check`
`SE`	a named vector of standard errors for each trait value; applied to all trait values if given a single value
`ngen`	number of sampling generations
`sample.freq`	frequency with which Markov samples are retained (e.g., `sample.freq=10` retains every tenth sample in the chain)
`prob.mergesplit`	proportion of proposals that split or merge rates, thereby altering complexity of the evolutionary model
`prob.root`	proportion of proposals in which the root state is updated
`lambdaK`	the shape parameter for the Poisson prior-distribution on number of distinct rates
`constrainK`	a constraint of model complexity (e.g., `constrainK=2` restricts sampling to models with two independent rates)
`jumpsize`	if `NULL`, the proposal width is calibrated; otherwise, proposal width is constrained to the supplied value

`simplestart` if `FALSE`, a random model complexity is chosen; if `TRUE`, Markov sampling begins under a global-rate model

`internal.only` dictates whether any branch in the tree can be chosen for a rate shift or tips only (`internal.only=FALSE`)

`summary` a logical switch that determines whether any output is printed to the console after the analysis has finished

`fileBase` a string used to uniquely identify the base directory to which Markov samples are written

Details

In the two primary objects that store posterior estimates of `shifts` and `rates` (see **Value**), branches are labeled in accordance with the numeric identifiers found in `phy$edge`, which represents a table of ancestor-descendant relationship (see `read.tree`. In the **Examples** below do **not** expect convergence with such short chains!

Value

If `summary=TRUE` the global rate of acceptance and calibrated proposal width is returned as a list. After a run has completed, the rates of acceptance for various proposal mechanisms are printed, along with the (calibrated) jumpsize used for chain sampling.

Posterior results are written to several files within a base directory, the contents of which are as follows:

`logfile` a logfile including the following for each Markov sample; the generations at which samples were retained; the generalized model of trait evolution used for inference (e.g., Brownian motion or 'BM"); the mean treewide rate in the sampled generation; the number of independent rates; the root state; and the likelihood of the model at each sample. Do not be alarmed if it seems to take some time before values begin to appear in the `logfile`: samples are not saved until after the calibration period has terminated

`errors` a logfile to which errors are recorded, if any are generated in the Markov chain

`posteriorsamples` a compressed .rda file, storing estimates from the Markov chain for relevant parameters; contents of this file can be loaded with `load("posteriorsamples.rda")`. This list object, whose referent is also `posteriorsamples` when in R-memory, includes two dataframes: `rates` comprises the relative-rate estimates for each branch in the phylogeny; `rate.shifts` stores the branches that were chosen for rate shifts in each retained sample of the Markov chain (`1` signifies a shift; `0` implies a branch that was not sampled for a rate shift). Column names of both components of `posteriorsamples` are from the numeric vector of branch labels (see **Details**). For further information on .Rd and .rda files, see `save` and `load` as well as the **Examples** below.

Author(s)

LJ Harmon, AL Hipp, and JM Eastman

Examples

```
#############
## generate tree
n=24
while(1) {
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))
phy$tip.label=paste("sp",1:n,sep="")
rphy=reorder(phy,"pruningwise")

# find an internal edge
anc=get.desc.of.node(Ntip(phy)+1,phy)
branches=phy$edge[,2]
branches=branches[branches>Ntip(phy) & branches!=anc]
branch=branches[sample(1:length(branches),1)]
desc=get.descendants.of.node(branch,phy)
if(length(desc)>=4) break()
}
rphy=phy
rphy$edge.length[match(desc,phy$edge[,2])]=phy$edge.length[match(desc,phy$edge[,2])]*64

e=numeric(nrow(phy$edge))
e[match(c(branch,desc),phy$edge[,2])]=1
cols=c("red","gray")
dev.new()
plot(phy,edge.col=ifelse(e==1,cols[1],cols[2]), edge.width=2)
mtext("expected pattern of rates")

#############
## simulate data on the 'rate-shifted' tree
dat=rTraitCont(phy=rphy, model="BM", sigma=sqrt(0.1))

## run two short reversible-jump Markov chains
r=paste(sample(letters,9,replace=TRUE),collapse="")
lapply(1:2, function(x) rjmcmc.bm(phy=phy, dat=dat, ngen=10000, sample.freq=10, prob.merg

 # collect directories
dirs=dir("./",pattern=paste("BM",r,sep="."))
pool.rjmcmcsamples(base.dirs=dirs, lab=r)

## view contents of .rda
load(paste(paste(r,"combined.rjmcmc",sep="."),paste(r,"posteriorsamples.rda",sep="."),sep
print(head(posteriorsamples$rates))
print(head(posteriorsamples$rate.shifts))

## plot Markov sampled rates
dev.new()
shifts.plot(phy=phy, base=paste(r,"combined.rjmcmc",sep="."), burnin=0.5, legend=TRUE, pd

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`rjmcmc.data` *example datasets for auteur*

Description

Provides example body-size data and a corresponding phylogeny

Usage

```
data(primates); data(chelonia); data(urodela)
```

Details

Each object is a list of two items, a `phy` object and a `dat` object. The `phy` object is a phylogenetic tree of class 'phylo' (see `read.tree`). The `dat` object is a named vector of body sizes for each group.

References

Data are from the following sources:

PRIMATES

- REDDING DW, C DEWOLFF, and AO MOOERS. 2010. Evolutionary distinctiveness, threat status and ecological oddity in primates. Conservation Biology 24:1052-1058.
- VOS RA and AO MOOERS. 2006. A new dated supertree of the Primates. Chapter 5. In: VOS RA (Ed.) Inferring large phylogenies: the big tree problem. [Ph.D. thesis]. Burnaby BC, Canada: Simon Fraser University.

TURTLES

- JAFFE, G SLATER, and M ALFARO. in review, Evolution. Ecological habitat and body size evolution in turtles.

SALAMANDERS

- ADAMS DC, CM BERNS, KH KOZAK, and JJ WIENS. 2009. Are rates of species diversification correlated with rates of morphological evolution? Proceedings of the Royal Society of London B 276:2729-2738.
- BONETT RM, PT CHIPPINDALE, PE MOLER, RW VAN DEVENDER, and DB WAKE. 2009. Evolution of gigantism in amphiumid salamanders. PLoSONE 4(5):e5615.
- EASTMAN JM and A STORFER. in review, Nature Communications. Hybridism swallows salamander diversity.
- KOZAK KH, RW MENDYK, and JJ WIENS. 2009. Can Parallel Diversification Occur in Sympatry? Repeated Patterns of Body-Size Evolution in Coexisting Clades of North American Salamanders. Evolution 63:1769-1784.
- WEISROCK DW, TJ PAPENFUSS, JR MACEY, SN LITVINCHUK, R POLYMENI, IH UGURTAS, E ZHAO, H JOWKAR, and A LARSON. 2006. A molecular assessment of phylogenetic relationships and lineage accumulation rates within the family Salamandridae (Amphibia, Caudata). Molecular Phylogenetics and Evolution 41:368-383.
- WIENS JJ and JT HOVERMAN. 2008. Digit reduction, body size, and paedomorphosis in salamanders. Evolution and Development 10:449-463.
- ZHANG P, Y-Q CHEN, H ZHOU, X-L WANG, TJ PAPENFUSS, DB WAKE and L-H QU. 2006. Phylogeny, evolution, and biogeography of Asiatic salamanders (Hynobiidae). Proceedings of the National Academy of Sciences USA 103:7360-7365.
- ZHANG P and DB WAKE. 2009. Higher-level salamander relationships and divergence dates inferred from complete mitochondrial genomes. Molecular Phylogenetics and Evolution. 53:492-508.

rjmcmc.ou *Bayesian sampling of shifts in trait optima: Ornstein-Uhlenbeck process*

Description

Implements reversible-jump Markov chain Monte Carlo sampling to identify shifts in the process of continuous trait evolution across phylogenetic trees

Usage

```
rjmcmc.ou(phy, dat, SE = 0, ngen = 1000, sample.freq = 100, prob.mergesplit = 0.
```

Arguments

`phy`	a phylogenetic tree of class 'phylo'
`dat`	a named vector of continuous trait values, associated with each species in `phy`; see `name.check`
`SE`	a named vector of standard errors for each trait value; applied to all trait values if given a single value
`ngen`	number of sampling generations
`sample.freq`	frequency with which Markov samples are retained (e.g., `sample.freq=10` retains every tenth sample in the chain)
`prob.mergesplit`	proportion of proposals that split or merge optima, thereby altering complexity of the evolutionary model
`prob.theta`	proportion of proposals in which selective regimes for branches are updated
`prob.alpha`	proportion of proposals in which the evolutionary constraint parameter is updated
`upper.alpha`	an upper bound on the evolutionary constraint parameter
`lambdaK`	the shape parameter for the Poisson prior-distribution on number of distinct optima
`constrainK`	a constraint of model complexity (e.g., `constrainK=2` restricts sampling to models with two independent optima)
`jumpsize`	if `NULL`, the proposal width is calibrated; otherwise, proposal width is constrained to the supplied value
`simplestart`	if `FALSE`, a random model complexity is chosen; if `TRUE`, Markov sampling begins under a global-optimum model
`internal.only`	dictates whether any branch in the tree can be chosen for a regime shift or tips only (`internal.only=FALSE`)
`summary`	a logical switch that determines whether any output is printed to the console after the analysis has finished
`fileBase`	a string used to uniquely identify the base directory to which Markov samples are written

Details

In the two primary objects that store posterior estimates of `shifts` and `optima` (see **Value**), branches are labeled in accordance with the numeric identifiers found in `phy$edge`, which represents a table of ancestor-descendant relationship (see `read.tree`. In the **Examples** below do **not** expect convergence with such short chains!

Value

If `summary=TRUE` the global rate of acceptance and calibrated proposal width is returned as a list. After a run has completed, the rates of acceptance for various proposal mechanisms are printed, along with the (calibrated) jumpsize used for chain sampling.

Posterior results are written to several files within a base directory, the contents of which are as follows:

`logfile` a logfile including the following for each Markov sample; the generations at which samples were retained; the generalized model of trait evolution used for inference (e.g., Brownian motion or 'BM"); the mean treewide optimum trait value in the sampled generation; the number of independent optima; and the likelihood of the model at each sample. Do not be alarmed if it seems to take some time before values begin to appear in the `logfile`: samples are not saved until after the calibration period has terminated

`errors` a logfile to which errors are recorded, if any are generated in the Markov chain

`posteriorsamples`
a compressed .rda file, storing estimates from the Markov chain for relevant parameters; contents of this file can be loaded with `load("posteriorsamples.rda")`. This list object, whose referent is also `posteriorsamples` when in R-memory, includes two dataframes: `optima` comprises the inferred selective regime (trait optimum) for each branch in the phylogeny; `shifts` stores the branches that were chosen for shifts in selective regime for each retained sample of the Markov chain (`1` signifies a shift; `0` implies a branch that was not sampled for a regime shift). Column names of both components of `posteriorsamples` are from the numeric vector of branch labels (see **Details**). For further information on .Rd and .rda files, see `save` and `load` as well as the **Examples** below.

Note

This function is largely based on code from `ouch-package`

Author(s)

AA King and JM Eastman

Examples

```
#############
## generate tree
n=24
while(1) {
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))
phy$tip.label=paste("sp",1:n,sep="")
rphy=reorder(phy,"pruningwise")

# find an internal edge
```

```
anc=get.desc.of.node(Ntip(phy)+1,phy)
branches=phy$edge[,2]
branches=branches[branches>Ntip(phy) & branches!=anc]
branch=branches[sample(1:length(branches),1)]
desc=get.descendants.of.node(branch,phy)
if(length(desc)>=4) break()
}
rphy=phy

## build array of optima
t=numeric(nrow(phy$edge))
t[match(c(branch,desc),phy$edge[,2])]=8
cols=c("red","gray")
dev.new()
plot(phy,edge.col=ifelse(t==8,cols[1],cols[2]), edge.width=2)
mtext("expected pattern of optima")

#############
## simulate data on the 'optimum-shifted' tree: shift from optimum 0.0 to optimum 2.0
dat=rTraitCont(phy=rphy, model="OU", sigma=sqrt(0.01), alpha=0.1, theta=t)

## run two short reversible-jump Markov chains
r=paste(sample(letters,9,replace=TRUE),collapse="")
lapply(1:2, function(x) rjmcmc.ou(phy=phy, dat=dat, ngen=10000, sample.freq=10, prob.merg

 # collect directories
dirs=dir("./",pattern=paste("OU",r,sep="."))
pool.rjmcmcsamples(base.dirs=dirs, lab=r)

## plot Markov sampled optima
dev.new()
shifts.plot(phy=phy, base=paste(r,"combined.rjmcmc",sep="."), burnin=0.5, legend=TRUE, pc

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`shifts.plot` *plotting of Bayesian posterior samples*

Description

Plots a phylogeny with topological indication of posterior support for evolutionary shifts

Usage

```
shifts.plot(phy, base.dir, burnin = 0, level = 0.01, internal.only = FALSE, pain
```

Arguments

`phy`	a phylogenetic tree of class 'phylo'
`base.dir`	a directory name, in which the results from `rjMCMC` are to be found
`burnin`	proportion of the chain to be treated as burnin (e.g., from 0 to 1)

`level` a level of posterior support (from 0 to 1) for a branch-associated evolutionary shifts, above which results will be compiled

`internal.only` dictates whether tips are excluded from processing of evolutionary shifts (`internal.only=TRUE`)

`paint.branches` a logical switch governing whether branches in the phylogeny are to be shaded according to their model-averaged estimates

`legend` a logical switch that determines whether keys for the generated symbols and colors are plotted

`pdf` a logical switch that determines whether a .pdf of the plot is written to the 're-sults' subdirectory of the stored Bayesian output

`verbose` a logical switch that determines whether additional diagnositics are generated for all branches receiving posterior support beyond the defined `level`; if `verbose=TRUE`, results will be stored in a 'results' subdirectory

`lab` a string used to uniquely identify the .pdf if created

`...` additional arguments to be passed to `plot.phylo`

Value

Returns, as a side-effect, a plotted phylogenetic tree with colorized summary of posterior estimates. Colors that strongly differ from gray are lower (blue) or higher (red) than the median posterior estimate across the tree. Bubbles at branches are proportional to posterior support for a shift occurring at the indicated branch, conditioned on the underlying evolutionary process involving at least one shift.

Author(s)

JM Eastman

Examples

```
# generate tree
n=24
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))

#############
## simulate a rate shift
 # find an internal edge
anc=get.desc.of.node(Ntip(phy)+1,phy)
branches=phy$edge[,2]
branches=branches[branches>Ntip(phy) & branches!=anc]
branch=branches[sample(1:length(branches),1)]
desc=get.descendants.of.node(branch,phy)
rphy=phy
rphy$edge.length[match(desc,phy$edge[,2])]=phy$edge.length[match(desc,phy$edge[,2])]*64
e=numeric(nrow(phy$edge))
e[match(c(branch,desc),phy$edge[,2])]=1
cols=rev(diverge_hcl(n=2))
dev.new()
plot(phy,edge.col=ifelse(e==1,cols[1],cols[2]), edge.width=2)
mtext("expected pattern of rates")
```

```
## simulate data on the 'rate-shifted' tree
dat=rTraitCont(phy=rphy, model="BM", sigma=sqrt(0.1))

## run two short reversible-jump Markov chains
r=paste(sample(letters,9,replace=TRUE),collapse="")
lapply(1:2, function(x) rjmcmc.bm(phy=phy, dat=dat, ngen=10000, sample.freq=10, prob.merg

 # collect directories
dirs=sapply(1:2, function(x) dir("./",pattern=paste("BM",r,x,sep=".")))
pool.rjmcmcsamples(base.dirs=dirs, lab=r)

## plot Markov sampled rates
dev.new()
shifts.plot(phy=phy, base=paste(r,"combined.rjMCMC",sep="."), burnin=0.5, legend=TRUE, po

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`trace.plot` *plotting of Bayesian posterior samples*

Description

Compares posterior densities of Bayesian estimates

Usage

```
trace.plot(obj, col, alpha, lwd = 1, hpd = 0.95, bars = TRUE, legend.control = l
```

Arguments

`obj`	a vector or dataframe (where each column is separately plotted); names of columns are retrieved if a legend is to be generated
`col`	either empty or a vector of colors with one for each column in the `obj`
`alpha`	a degree of color translucence to be used for shaded densities, where 1 is opaque, 0 is clear
`lwd`	line width used for highest-density range and density outlines (see `par`)
`hpd`	either `NULL` or a value between 0 and 1, corresponding to the width of the highest density region to be shaded
`bars`	a logical specifier which, if `bars=TRUE`, outlines the width of the high density region
`legend.control`	if `plot=TRUE`, a legend is generated (see `legend` for details concerning this list)
`truncate`	if `min` and (or) `max` are defined, values below and (or) above the truncation point are removed prior to computation of the highest density region and prior to plotting
`xlim`	if `min` and (or) `max` are defined, values below and (or) above the limit(s) are excluded from the plot without further corrupting the data
`...`	further arguments to be passed to `plot`

Author(s)

JM Eastman

See Also

profiles.plot, upon which this function is largely based

Examples

```
# construct and plot a dataframe of three 'traces', excluding the largest values
d=data.frame(list(rnorm(n=1000,mean=9,sd=5)),list(rnorm(n=1000,mean=1.5,sd=2)),list(rnorm
names(d)=letters[1:3]
trace.plot(obj=d, xlim=list(max=10), col=c("maroon","gray","purple"), alpha=0.3, lwd=2)

dev.new()
# construct and plot a dataframe of two 'traces' using a log scale
d=data.frame(list(rlnorm(n=1000,meanlog=1,sdlog=0.5)),list(rlnorm(n=1000,meanlog=1.5,sdlo
names(d)=letters[1:2]
trace.plot(obj=d,col=c("maroon","purple"),alpha=0.3,log="x")
```

---

tracer — *plotting of Bayesian posterior samples*

Description

Generates diagnostics for reversible-jump Markov sampling densities

Usage

```
tracer(base.log, lambdaK = log(2), Bayes.factor = NULL, burnin = 0.25, col.line
```

Arguments

`base.log`	a path to the stored summary logfile of the Markov sample
`lambdaK`	the shape parameter for the Poisson distribution used as a prior on the number of distinct evolutionary parameters in the tree
`Bayes.factor`	a vector of two numbers, specifying a Bayes factor comparison between `j`- and `k`-parameter models
`burnin`	proportion of the chain to be treated as burnin (e.g., from 0 to 1); burnin is used for all plots but the trace of the log-likelihoods
`col.line`	a color to be used for density plots (both for a `Bayes.factor` comparison and for the density of mean rates across the Markov sample)
`pdf`	a logical switch that determines whether plots are written to .pdf
`factor`	a character `k, M` or `NULL`, determining whether the lnL-trace axis is in thousands (`k`) or millions (`M`) of generations
`...`	further arguments to be passed to `plot`

Value

if `pdf=TRUE`, a .pdf will be generated that includes the trace of log-likelihoods, the density of sampled mean rates, the correspondence between prior and posterior weights for `k`-rate models, and a Bayes factor comparison. The argument for `Bayes.factor` can be `Bayes.factor=c(1,"multi")` in which a comparison between a global-parameter and multiple-parameter model is made

Author(s)

JM Eastman

Examples

```
# generate tree
n=24
phy=prunelastsplit(birthdeath.tree(b=1,d=0,taxa.stop=n+1))

# simulate data
dat=rTraitCont(phy=phy, model="BM", sigma=sqrt(0.1))

# run reversible-jump MCMC for a short chain
r=paste(sample(letters,9,replace=TRUE),collapse="")
rjmcmc.bm(phy=phy, dat=dat, ngen=5000, sample.freq=10, prob.mergesplit=0.1, fileBase=r)

# plot Markov sampled rates and other traces
tracer(base.log=paste(paste("BM",r,"parameters",sep="."),paste("BM",r,"rjmcmc.log",sep=".

## PASTE UNCOMMENTED FOLLOWING LINE TO DROP DIRECTORIES CREATED BY RJMCMC
 # unlink(dir(pattern=paste(r)),recursive=TRUE)
```

`vmat` *computation of phylogenetic variance-covariance matrix*

Description

Calculates the VCV matrix for a phylogenetic tree

Usage

```
vmat(phy)
```

Arguments

`phy`	a phylogenetic tree of class 'phylo'

Details

This function is a conversion of `vcv.phylo` into compiled `C++` for rapid generation of the expected trait-variances and trait covariances among species under Brownian motion evolution. This function is efficient for trees with fewer than ca. 5000 tips; for larger trees, sufficient memory (> 2 Gb RAM) may be required.

Value

A variance-covariance matrix for all tips within the supplied tree.

Author(s)

JM Eastman, based on `vcv.phylo` by Emmanuel Paradis

Examples

```
## generate tree
n=250
phy=rescaleTree(phy=rcoal(n=n),totalDepth=100)

## compare function times for vcv.phylo() and vmat()
print(system.time(vcv.phylo(phy)))
print(system.time(vmat(phy)))

## generate some smaller matrices
n=4
phy=rescaleTree(phy=rcoal(n=n),totalDepth=100)

## compute the variance-covariance matrix with ape and rjmcmc
vcv.phylo(phy)->vAPE
vmat(phy)->vRJ

## print the matrices
print(vAPE)
print(vRJ)

## verify that both packages return identical results
print(all(vAPE==vRJ))
```

Index