# Package ‘geiger’

March 1, 2011

**Type** Package

**Title** Analysis of evolutionary diversification

**Version** 1.3-2

**Date** 2011-02-09

**Author** Luke Harmon `<lukeh@uidaho.edu>`, Jason Weir, Chad Brock, Rich Glor, Wendell Challenger, Gene Hunt

**Maintainer** Luke Harmon `<lukeh@uidaho.edu>`

**Depends** ape (>= 1.9-4), MASS, mvtnorm, msm, ouch, Rcpp

**LinkingTo** Rcpp

**Description** Running macroevolutionary simulation, and estimating parameters related to diversification from comparative phylogenetic data.

**License** GPL (>= 2)

**URL** `http://www.webpages.uidaho.edu/~lukeh/index.html`

**Repository** CRAN

**Date/Publication** 2009-10-20 09:33:19

**Archs** i386, x86_64

## R topics documented:

## `geiger-package` *GEIGER*

### Description

A package for macroevolutionary simulation and estimating parameters related to diversification from comparative phylogenetic data.

### Details

| | |
|---|---|
| Package: | geiger |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2007-01-19 |
| License: | GPL version 2 or greater? |

### Author(s)

Luke J. Harmon, Jason Weir, Chad Brock, and Wendell Challenger

Maintainer: Luke Harmon <lukeh@uidaho.edu>

### Examples

```
# Disparity-through-time, as in Harmon et al. 2003
data(geospiza)
attach(geospiza)

dtt.full(geospiza.tree, geospiza.data)


# Simulation tester - verifies that simulations produce expected vcv structure
sims<-sim.char(geospiza.tree, as.matrix(1.0), n=100000)
m<-var(t(sims[,1,]))
m2<-vcv.phylo(geospiza.tree)
round(m-m2)
# Equal to zero because the vcv matrices are as expected.
```

```
#Test models of evolution
fitContinuous(geospiza.tree, geospiza.data)


# Reconstruct evolutionary vcv matrix
v<-cbind(c(4.0, -1.5),c(-1.5, 9.0))
n<-1000
sims<-sim.char(geospiza.tree, v, n)
r<-array(dim=c(2, 2, n))
for(i in 1:n)
r[,,i]<-ic.sigma(geospiza.tree, sims[,,i])
v.recon<-apply(r, c(1,2), mean)
v-v.recon

# Discrete character evolution
q<-list(rbind(c(-.01, .01), c(.01, -.01)))
sims<-sim.char(geospiza.tree, q, model="discrete", n=100)
```

---

## `area.between.curves` *Area between two curves*

---

### Description

Finds the area between two curves (f1 and f2) for a given range on the x-axis. One use for this function is to calculate the MDI statistic from Harmon et al. 2003

### Usage

```
area.between.curves(x, f1, f2, xrange = c(0,1))
```

### Arguments

| | |
|---|---|
| `x` | Vector of values for the x axis |
| `f1` | Y values for curve 1 |
| `f2` | Y values for curve 2 |
| `xrange` | Range of x values over which area is calculated |

### Details

Area is calculated between the two curves f1 and f2 for x values in the given range. If f2 is greater than f1, areas are positive; otherwise, areas are negative

### Value

Area between f1 and f2 between 0 and cutoff

### Author(s)

Luke J. Harmon

### References

Harmon, L. J., J. A. Schulte, J. B. Losos, and A. Larson. 2003. Tempo and mode of evolutionary radiation in iguanian lizards. Science 301: 961-964.

### Examples

```
data(geospiza)
attach(geospiza)

gg<-dtt.full(geospiza.tree, geospiza.data)

# Full area, as in dtt.full function
area.between.curves(gg$times, apply(gg$dtt.sims,1,mean), gg$dtt.data)

# Area for first 2/3, as in Harmon et al. 2003
area.between.curves(gg$times, apply(gg$dtt.sims,1,median), gg$dtt.data, xrange=c(0, 2/3))

# You could also use the mean
area.between.curves(gg$times, apply(gg$dtt.sims,1,mean), gg$dtt.data, xrange=c(0, 2/3))
```

---

## `BDsim` *Birth-death population simulator*

---

### Description

Simulates species richness (or population growth) under a uniform, time-homogeneous birth-death process.

### Usage

```
BDsim(nStart, b, d, times)
```

### Arguments

| | |
|---|---|
| `nStart` | Number of taxa at starting time zero |
| `b` | Per-lineage birth (speciation) rate |
| `d` | Per-lineage death (extinction) rate |
| `times` | Vector of times where extant species are counted |

### Details

This function simulates species diversification under a uniform birth-death process. This differs from birthdeath.tree in that only the number of species, and not their phylogenetic affinities, are stored. This function relates to GEIGER's rate.estimate (and associated functions), which are also non-phylogenetic.

### Value

Population size at each time in TIMES

### Author(s)

Richard E. Glor and Luke J. Harmon

### References

Yule, G. U. 1924. A mathematical theory of evolution based on the conclusions of Dr. J. C. Willis, FRS. Philos. Trans. R. Soc. London Ser. B 213: 21-87

### See Also

rate.estimate, stem.p, crown.p, stem.limits, crown.limits

### Examples

```
pop1=BDsim(nStart=10, b=0.1, d=0, times=1:10)
pop2=BDsim(nStart=10, b=0, d=0.1, times=1:10)
pop3=BDsim(nStart=10, b=0.1, d=0.1, times=1:10)

plot(pop1, type="l", ylim=c(0,20))
lines(pop2, col="red")
lines(pop3, col="blue")
```

---

## `birthdeath.tree` *Birth-death tree simulator*

---

### Description

Simulates phylogenetic trees under a uniform birth-death process.

### Usage

```
birthdeath.tree(b, d, time.stop = 0, taxa.stop = 0, seed = 0, print.seed=FALSE,
```

### Arguments

| | |
|---|---|
| `b` | Per-lineage birth (speciation) rate |
| `d` | Per-lineage death (extinction) rate |
| `time.stop` | Stopping time |
| `taxa.stop` | Maximum number of taxa |
| `seed` | Random number seed; if seed=0 (default) then random number generator is seeded based on the clock |
| `print.seed` | If T, prints out the random number seed associated with each tree |
| `return.all.extinct` | Return trees where all lineages have gone extinct? Otherwise, the function will build trees until it gets one with at least one surviving lineage. |

## Details

Starting from a root node - i.e., two living lineages - this function simulates the growth of a phylogenetic tree under a uniform, time-homogeneous birth-death process. This means that every lineage has a constant probability of speciating, and a constant probability of going extinct, per unit time. If birth is greater than death, then the number of lineages is expected to grow exponentially.

## Value

Phylogenetic tree in ape format. If death rate is non-zero, then the returned tree will likely include some extinct lineages (terminating before the present day). The GEIGER function prune.extinct.taxa can remove these lineages.

## Note

One note of caution: it is easy to set parameter values that result in tremendously HUGE trees. If the function seems to hang up, this could be the problem.

## Author(s)

Luke J. Harmon and Jason Weir

## References

Geiger

## See Also

BDsim for non-phylogenetic simulations; prune.extinct.taxa

## Examples

```
# Pure-birth tree
p1<-birthdeath.tree(b=0.1, d=0, time.stop=20)
plot(p1)

# Birth-death tree with extinct taxa
# The return.all.extinct flag prevents trees with no survivors

p2<-birthdeath.tree(b=0.2, d=0.05, time.stop=20, return.all.extinct=FALSE)
plot(p2)

# Previous tree with extinct taxa removed
prune.extinct.taxa(p2)->p3
```

## `deltaTree` *Tree transformations*

### Description

Apply various transformation to the branches of a phylogenetic tree.

### Usage

```
deltaTree(phy, delta, rescale = T)
lambdaTree(phy, lambda)
kappaTree(phy, kappa)
ouTree(phy, alpha)
tworateTree(phy, breakPoint, endRate)
linearchangeTree(phy, endRate=NULL, slope=NULL)
exponentialchangeTree(phy, endRate=NULL, a=NULL)
speciationalTree(phy)
rescaleTree(phy, totalDepth)
```

### Arguments

| | |
|---|---|
| `phy` | an object of class phylo |
| `delta` | Delta value |
| `rescale` | if TRUE, rescale tree so that the total depth (root-to-tip distance) remains constant; this facilitates comparisons of rates among different delta parameter values |
| `lambda` | Lambda value |
| `kappa` | Kappa value |
| `alpha` | Value of Ornstein-Uhlenbeck constraint parameter |
| `breakPoint` | Applies a new rate of evolution after the time specified as the breakpoint |
| `endRate` | Rate of change at the present day, relative to the initial (root) rate |
| `slope` | Slope of the relationship between rate and time in the linearchange model |
| `a` | Exponent of the relationship between rate and time in the exponentialchange model |
| `totalDepth` | Desired value for root to tip distance of the tree. |

### Details

All of these functions take a tree and various parameters as inputs, and output a new tree structure that has been transformed in the appropriate manner. These are meant to correspond with changing the model of phenotypic evolution for discrete or continuous characters. To use the modified model, one then evolves characters on the transformed tree under a Brownian motion model. Another use for these functions is in finding the particular parameter values that maximize the likelihood of the given data; they can then be compared to Brownian motion using a likelihood-ratio test, or to any other model using AIC. deltaTree: A transformation suggested by Pagel as a test for a slow-down or speed-up in the rate of character evolution through time. Value of delta > 1 disproporionately increase the length of external nodes (speed-up), where as values of delta < 1 disproportionately increase the length of internal nodes (slow-down). Delta = 1 is a Brownian motion model (so the tree is returned unchanged). lambdaTree: Another Pagel transformation. This one manipulates the tree as a test of phylogenetic signal. Values of lambda should always less than one. Each internal edge is multiplied by lambda, while branches leading to tips are unchanged; this has the effect of reducing or eliminating phylogenetic signal. Lambda = 1 is a Brownian motion model (so the tree is returned unchanged). kappaTree: A third Pagel transformation. This one manipulates the tree as a test of "speciational" models. Each branch length in the tree is raised to the power kappa. Kappa = 1 is a Brownian motion model (so the tree is returned unchanged), while kappa = 0 is a speciational model where all branch lengths are equal. Kappa > 1 or < 0 does not make sense to me, so I think the values should be restricted to between zero and one. ouTree: A transformation that corresponds to evolution under an Ornstein-Uhlenbeck model. This model can be thought of as a random-walk model with a central tendency, so that phynotypes tend to evolve towards one "optimal" value. Evolving characters on the tree that's returned from this function mimics evolving them under the OU model in the case where the optimal value is exactly equal to the ancestral state for the character. If these two values are NOT equal, then simply transforming the tree is not enough to run simulations - and I haven't implemented any other way of doing this. tworateTree: A transformation that effectively changes the rate of evolution at some point in time, specified by breakPoint. The rate after the breakpoint is supplied as endRate, and only needs to be specified as a ratio relative to the initial rate. Thus, if endRate is greater than 1, evolution speeds up (i.e. all branches towards the tips are made longer; if endrate is less than 1, branches towards the tips are made shorter. If endRate = 1, the model is a constant- rate Brownian motion model (so the tree is returned unchanged). linearchangeTree: A transformation that changes the rate of evolution linearly through time. The rate at the present day is supplied either as endRate, a ratio relative to the initial rate at the root of the tree, or as the slope of the relationship between rate and time. Rates change linearly: r(t) = ro + slope * t, where ro is the inital rate and a is the slope determined using endRate. If endRate is greater than 1 or slope > 0, evolution gradually speeds up; if endrate is less than 1 or slope < 0, evolution gradually slows down. If endRate = 1, the model is a constant- rate Brownian motion model (so the tree is returned unchanged). exponentialchangeTree: A transformation that changes the rate of evolution exponentially through time. The rate at the present day is supplied as endRate or a, in exactly the same way as in linearchangeTree. The difference for this function is that rates grow or decay exponentially rather than linearly. The equation is r(t) = ro * exp(a * t), where ro is the inital rate and a either given or is determined using endRate. One of the two (a or endRate) must be specified, or the function returns an error. speciationalTree: Transforms all the branch lengths in the tree to a length of one; sometimes this is called a 'speciational' or 'punctuated' model of evolution. rescaleTree: Rescales a tree so that the total depth of the tree (i.e. the distance from the root to the tips) equals totalDepth. Some of the functions from Pagel, above, change the total depth of the tree, so comparing parameters estimated under these models can be difficult.

## Value

Transformed phylogenetic tree, object of class phylo (ape format).

## Author(s)

Luke J. Harmon

## References

Delta and lambda: Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature 401:877-884. OU: Butler, M.A. and A.A. King, 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695. Others: Various papers in prep., L. J. Harmon and J. T. Weir.

## Examples

```
data(geospiza)
attach(geospiza)

deltaTree(geospiza.tree, 0.5)->g2
plot(g2)
lambdaTree(geospiza.tree, 0.5)->g3
plot(g3)
kappaTree(geospiza.tree, 0.5)->g3b
plot(g3b)
tworateTree(geospiza.tree, 0.5, 0.5)->g4
plot(g4)
linearchangeTree(geospiza.tree, 0.1)->g5
plot(g5)
exponentialchangeTree(geospiza.tree, 0.1)->g5
plot(g5)
speciationalTree(geospiza.tree)->g6
plot(g6)
rescaleTree(geospiza.tree, 100)->g7
plot(g7)
```

---

## dtt *Disparity-through-time*

---

### Description

Functions for calculating and plotting disparity-through-time for a phylogenetic tree and phenotypic data.

### Usage

```
disp.calc(data, disp = "avg.sq")
dtt(phy, data, data.names=NULL, disp = "avg.sq")
dtt.full(phy, data, data.names=NULL, disp="avg.sq", nsims=1000, mdi.range=c(0,1)
```

### Arguments

| | |
|---|---|
| `data` | Data matrix - either actual or simulated |
| `data.names` | Tip names for data vector that match tree species; ignored if data includes names |
| `phy` | Phylogenetic tree in 'phylo' format |
| `disp` | Disparity measure: see below for currently implemented options. |
| `nsims` | Number of simulations used to calculate null dtt plot |
| `mdi.range` | Time range over which to calculate MDI statistic (area between curves). Time is relative to the total tree length of 1; default is the whole tree from 0 (root) to 1 (tips) |

## Details

The most complete function, dtt.plot, carries out the entire disparity-through-time procedure described in Harmon et al. 2003. Other functions are for various parts of this, as follows: disp.calc: Calculate morphological disparity for a set of species. Disparity measure can be one of the following: avg.sq: Average squared euclidean distance among all pairs of points. This is a good choice if all of the axes are in the same units, or PC axes. avg.manhattan: Average Manhattan distance among all pairs of points. This is a good choice if the axes are all in different units, like colors and lengths. nb.states: Number of unique character states; this is the only option for discrete character data, for now.

dtt: Evaluates disparity-through-time for either a single data set or multiple data sets (for example, from simulations). dtt.full: Calculates dtt for the actual data, runs simulations using univariate or multivariate Brownian motion, calculates dtt for all simulations and finds the mean, and plots dtt for both data and simulations on the output device.

## Value

disp.calc: Disparity of the supplied data dtt: Average disparity for clades whose stem crosses each time interval in the tree. dtt.full: A list with the following items: dtt.data: DTT for the data dtt.sims: DTT for each simulated data set times: Times for each value in the dtt plot; this is just the branching times of the phylogeny MDI: Value of the MDI statistic, which is the area between the DTT plot for the data and the mean of the simulations ' Plot: Creates a DTT plot

## Author(s)

Luke J. Harmon

## References

Foote, M. 1997. The evolution of morphological diversity. Annual Review of Ecology and Systematics 28:129-152. Harmon, L. J., J. A. Schulte, J. B. Losos, and A. Larson. 2003. Tempo and mode of evolutionary radiation in iguanian lizards. Science 301: 961-964.

## Examples

```
data(geospiza)
attach(geospiza)

disp.calc(geospiza.data)
disp.data<-dtt(geospiza.tree, geospiza.data)
full.output<-dtt.full(geospiza.tree, geospiza.data)
```

---

`fitContinuous` *Model fitting for continuous data*

---

## Description

Fits macroevolutionary models to phylogenetic trees

## Usage

```
fitContinuous(phy, data, data.names=NULL, model=c("BM", "OU", "lambda", "kappa",
```

## Arguments

| | |
|---|---|
| `phy` | object of type phylo |
| `data` | Data vector (one trait) or matrix (multiple traits) |
| `data.names` | Tip names for data vector that match tree species; ignored if data includes names |
| `model` | Model to fit to comparative data; see below for options |
| `bounds` | Range to constrain estimates; see below for details |
| `meserr` | Measurement error for each trait for each tip species; can also be a single vector (if so, error is assumed to be equal for all traits) or a single number (error assumed equal for all traits across all species). |

## Details

This function fits various likelihood models for continuous character evolution. The function returns parameter estimates, (approximate) confidence intervals based on the Hessian of the likelihood function, and the likelihood. Likelihood is maximized using the r function nlm. This is a purely univariate function at this point - if multivariate data are sent to the function, it will carry out calculations for each character independently. Possible models are as follows: model="BM": Brownian motion model="OU": Ornstein-Uhlenbeck; fits a model of random walk with a central tendency proportional to the parameter alpha. Also called the "hansen" model in ouch, although the way the parameters are fit is slightly different here. model="lambda": Pagel's lambda; multiplies all internal branches of the tree by lambda, leaving tip branches as their original length. model="kappa": Pagel's kappa; raises all branch lengths to the power kappa. As kappa approaches zero, the model becomes speciational. model="delta": Pagel's delta; raises all node depths to the power delta. If delta is less than one, evolution in concentrated early in the tree; delta > 1 concentrates evolution towards the tips.

model="EB": Early burst, also called the ACDC model. Set by the eb parameter; fits a model where the rate of evolution increases or decreases exponentially through time, under the model r(t) = ro * exp(r * t), where ro is the inital rate and r is the rate change parameter. The actual parameter estimated, endRate, is the proportion of the initial rate represented by the end rate. Bounds for the parameters in the likelihood search are sometimes necessary. This is because in some cases, the likelihood surface can have long flat ridges that cause the search to get "stuck." This is particularly common, in my experience, for the OU model. You can set bounds with the "bounds" parameter. For example, to set bounds on alpha, use bounds=list(alpha=c(0.0001, 10)). One can also set multiple bounds at once: bounds=list(alpha=c(0.0001, 10), beta=c(1, 100)). This function might work better than in previous versions of Geiger. model="white": White noise model; all species drawn from the same normal distribution (no phylogenetic signal). model="trend": Brownian motion with a trend.

## Value

Returns a matrix of parameter estimates, along with approximate standard errors and 95 Also returns the log-likelihood of the model (lnl).

## Author(s)

Luke J. Harmon and Wendell Challenger

## References

Lambda, kappa, and delta: Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature 401:877-884. OU: Butler, M.A. and A.A. King, 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683-695. Early burst: Paper in revision., L. J. Harmon et al.

## Examples

```
data(geospiza)
attach(geospiza)


#---- PRINT RESULTS
fitContinuous(geospiza.tree, geospiza.data)

#---- STORE RESULTS
brownFit <-  fitContinuous(geospiza.tree, geospiza.data)

aic.brown<-numeric(5)
for(i in 1:5) aic.brown[i]<-brownFit[[i]]$aic

#----------------------------------------------------
#   PHYLOGENETIC SIGNAL: FIT LAMBDA
#----------------------------------------------------

lambdaFit<-fitContinuous(geospiza.tree, geospiza.data, model="lambda")

# Compare likelihoods:

d.lambda<-numeric(5)
for(i in 1:5) d.lambda[i]=2*(lambdaFit[[i]]$lnl-brownFit[[i]]$lnl)

# Calculate p values assuming chi-squared distribution with 1 d.f.
p.lambda=pchisq(d.lambda, 1, lower.tail=FALSE)

aic.lambda<-numeric(5)
for(i in 1:5) aic.lambda[i]<-lambdaFit[[i]]$aic

#----------------------------------------------------
#    TIME PROPORTIONALITY: DELTA
#---------------------------------------------------

deltaFit<-fitContinuous(geospiza.tree, geospiza.data, model="delta")

# Compare likelihoods:

d.delta<-numeric(5)
for(i in 1:5) d.delta[i]=2*(deltaFit[[i]]$lnl-brownFit[[i]]$lnl)

# Calculate p values assuming chi-squared distribution with 1 d.f.
p.delta=pchisq(d.delta, 1, lower.tail=FALSE)

aic.delta<-numeric(5)
for(i in 1:5) aic.delta[i]<-deltaFit[[i]]$aic
```

```
#-----------------------------------------------------
#   SPECIATIONAL MODEL: KAPPA
#----------------------------------------------------

kappaFit<-fitContinuous(geospiza.tree, geospiza.data, model="kappa")

# Compare likelihoods:

d.kappa<-numeric(5)
for(i in 1:5) d.kappa[i]=2*(kappaFit[[i]]$lnl-brownFit[[i]]$lnl)

# Calculate p values assuming chi-squared distribution with 1 d.f.
p.kappa=pchisq(d.kappa, 1, lower.tail=FALSE)

aic.kappa<-numeric(5)
for(i in 1:5) aic.kappa[i]<-kappaFit[[i]]$aic

#-----------------------------------------------------
#   OU MODEL: ALPHA
#----------------------------------------------------

ouFit<-fitContinuous(geospiza.tree, geospiza.data, model="OU")

# Compare likelihoods:

d.ou<-numeric(5)
for(i in 1:5) d.ou[i]=2*(ouFit[[i]]$lnl-brownFit[[i]]$lnl)

# Calculate p values assuming chi-squared distribution with 1 d.f.
p.ou=pchisq(d.ou, 1, lower.tail=FALSE)

aic.ou<-numeric(5)
for(i in 1:5) aic.ou[i]<-ouFit[[i]]$aic

#-----------------------------------------------------
#   EARLY BURST MODEL: R
#----------------------------------------------------

ebFit<-fitContinuous(geospiza.tree, geospiza.data, model="EB")

# Compare likelihoods:

d.eb<-numeric(5)
for(i in 1:5) d.eb[i]=2*(ebFit[[i]]$lnl-brownFit[[i]]$lnl)

# Calculate p values assuming chi-squared distribution with 1 d.f.
p.eb=pchisq(d.eb, 1, lower.tail=FALSE)

aic.eb<-numeric(5)
for(i in 1:5) aic.eb[i]<-ebFit[[i]]$aic

#-----------------------------------------------------
#   COMPARE ALL MODELS
#----------------------------------------------------

# One way: use likelihood ratio test to compare all models to Brownian model
```

```
d.all<-cbind(d.lambda, d.delta, d.kappa, d.ou, d.eb)
p.all<-cbind(p.lambda, p.delta, p.kappa, p.ou, p.eb)

cat("Trait\tlambda\tdelta\tkappa\tou\teb\n")

for(i in 1:5) {
cat("Tr", i, "\t");
for(j in 1:5) {
cat(round(d.all[i,j],2));
if(p.all[i,j]<0.05) cat("*");
if(p.all[i,j]<0.01) cat("*");
if(p.all[i,j]<0.001) cat("*");
cat("\t");
}
cat("\n");
}

# Another way: use AIC

aic.all<-cbind(aic.brown, aic.lambda, aic.delta, aic.kappa, aic.ou, aic.eb)
foo<-function(x) x-x[which(x==min(x))]
daic<-t(apply(aic.all, 1, foo))

rownames(daic)<-colnames(geospiza.data)
colnames(daic)<-c("Brownian", "Lambda", "Delta", "Kappa", "OU", "EB")

cat("Table of delta-aic values; zero - best model\n")
print(daic, digits=2)
```

## fitDiscrete *Model fitting for discrete comparative data*

### Description

Fits macroevolutionary models to phylogenetic trees

### Usage

```
fitDiscrete(phy, data, model=c("ER", "SYM", "ARD"), treeTransform=c("none", "lam
```

### Arguments

| | |
|---|---|
| `phy` | object of type phylo |
| `data` | Data vector (one trait) or matrix (multiple traits) |
| `model` | One of ER, SYM, or ARD; see below |
| `treeTransform` | Model for transforming the tree; see below |
| `data.names` | Tip names for data vector that match tree species; ignored if data includes names |
| `plotlnl` | Plot likelihood surface? Works only with models of two parameters or fewer. |

| | |
|---|---|
| `qLimits` | Vector giving minimum and maximum values for rate parameter q; used ONLY for plotting. |
| `pLimits` | Vector giving minimum and maximum values for whichever tree transformation parameter you're using; used ONLY for plotting. |

## Details

This function fits various likelihood models for discrete character evolution. Likelihood is maximized using the r function nlm. All of the models are continuous-time Markov models of trait evolution (see Yang 2006 for a good general discussion of this type of model). The function can handle traits with any number of character states, under a range of models. The character model is specified by the "model" argument: ER: Equal-rates (Mk) model; all transitions occur at equal rates. SYM: Symmetric transitions are equal; that is, 0->1 occurs at the same rate as 1->0, which may differ from the transition rate between states 1 and 2. For a 2 state character, this model is equivalent to ER ARD: All rates different model; each rate is a separate parameter.

The function returns a rate matrix, Q, giving the transition rates among the characters. The diagonals of this matrix are zero; this is only because the rows of the matrix sum to zero.

The treeTransform argument allows you to test models where rates vary across the tree. Options are: none: Rates are constant through time lambda: Pagel's lambda; multiplies all internal branches of the tree by lambda, leaving tip branches as their original length. kappa: Pagel's kappa; raises all branch lengths to the power kappa. As kappa approaches zero, the model becomes speciational. delta: Pagel's delta; raises all node depths to the power delta. If delta is less than one, evolution in concentrated early in the tree; delta > 1 concentrates evolution towards the tips. exponentialChange: A model where the rate of evolution changes exponentially through time. The difference between this option and linearChange is that rates grow or decay exponentially rather than linearly. The equation is r(t) = ro * exp(a * t), where ro is the inital rate and a is the rate of rate change. linearChange: A model where the rate of evolution changes linearly through time. Rates change linearly: r(t) = ro + a * t, where ro is the inital rate and a is the slope determined using endRate. If endRate is greater than 1, evolution gradually speeds up; if endrate is less than 1, evolution gradually slows down. If endRate = 1, the model is a constant- rate model.

twoRate: A model that effectively changes the rate of evolution at some point in time to endRate. If endRate is greater than 1, evolution speeds up, all branches towards the tips are made longer, if endrate is less than 1, branches towards the tips are made shorter. If endRate = 1, the model is a constant- rate model.

## Value

Returns maximum likelihood value for q and selected parameters, along with the likelihood score. Sometimes, parameters are confounded!

## Author(s)

Luke J. Harmon and R. E. Glor

## References

Yang, Z. 2006. Computational Molecular Evolution. Oxford University Press: Oxford.

## Examples

```
data(geospiza)
attach(geospiza)
```

```
gb<-as.factor(geospiza.data[,1]>4.2)
names(gb)<-rownames(geospiza.data)

fitDiscrete(geospiza.tree, gb)
fitDiscrete(geospiza.tree, gb, treeTransform="lambda")
fitDiscrete(geospiza.tree, gb, treeTransform="delta")
fitDiscrete(geospiza.tree, gb, treeTransform="kappa")

fitDiscrete(geospiza.tree, gb, treeTransform="linearChange")
fitDiscrete(geospiza.tree, gb, treeTransform="exponentialChange")
fitDiscrete(geospiza.tree, gb, treeTransform="twoRate")
```

---

## `geospiza` *Data from Darwin's finches*

---

### Description

Phylogenetic tree and morphological data for Darwin's finches

### Usage

```
geospiza
```

### Format

Tree in ape format and data table

### Source

Dolph Schluter

### References

Geiger

---

## `getAncStates` *Estimate ancestral character states for continuous characters*

---

### Description

Estimates ancestral character states for continuous characters under a Brownian motion model

### Usage

```
getAncStates(x, phy)
```

### Arguments

| | |
|---|---|
| `x` | Data vector |
| `phy` | Phylogenetic tree in phylo format |

### Details

This function uses an algorithm to calculate Maximum-likelihood estimates of ancestral character states for continuous characters.

The ape function ace carries out a similar function but finds estimates by actually maximizing the likelihood across all ancestors simultaneously.

I have implemented an algorithm that is faster and sometimes more reliable especially for large trees.

### Value

Vector of ancestral character states at each node. Nodes are numbered following the order in phy edge

### Author(s)

Luke J. Harmon

### See Also

ace

### Examples

```
data(geospiza)
attach(geospiza)

tt<-drop.tip(geospiza.tree, "olivacea")
getAncStates(geospiza.data[,1], tt)
```

---

## `ic.sigma` *Estimate Evolutionary VCV Matrix*

---

### Description

Uses independent contrasts to estimate the evolutionary variance-covariance matrix for n quantitative characters.

### Usage

```
ic.sigma(phy, data, data.names=NULL)
```

### Arguments

| | |
|---|---|
| `phy` | Phylogenetic tree in 'phylo' format |
| `data` | Data matrix |
| `data.names` | Tip names for data vector that match tree species; ignored if data includes names |

### Details

Data for this function should be a matrix of continuously-valued variables.

### Value

Returns the estimated evolutionary variance-covariance matrix of the variables under a multivariate Brownian motion model. If you have n characters in your analysis, this will be an nxn matrix. Diagonal elements represent rate estimates for individual characters, while off-diagonal elements represent the estimated covariance between two characters.

### Author(s)

Luke J. Harmon

### References

Revell, L. J., L. J. Harmon, R. B. Langerhans, and J. J. Kolbe. 2007. A phylogenetic approach to determining the importance of constraint on phenotypic evolution in the neotropical lizard, Anolis cristatellus. Evolutionary Ecology Research 9: 261-282.

### Examples

```
data(geospiza)
attach(geospiza)

ic.sigma(geospiza.tree, geospiza.data)
```

---

## `name.check` *Compares taxa in data and tree*

---

### Description

This function is a general tool for checking for concordance between a data file and a phylogenetic tree. For the data, names can be specified as the names of objects in the vector, rownames of the data array or as 'data.names'. The name.check function finds and lists all taxa present in data set but not in the tree, and vice-versa. The treedata function returns a list containing both the tree and the data after pruning out any species that are not found in both.

### Usage

```
name.check(phy, data, data.names=NULL)
```

### Arguments

| | |
|---|---|
| `phy` | an object of class "phylo" |
| `data` | data for tips of the tree |
| `data.names` | names of the tips in the order of the data; if this is not given, names will be taken from the names or rownames of the object data |

### Value

| | |
|---|---|
| `Tree.not.data` | Taxa in tree but not data |
| `Data.not.tree` | Taxa in data but not tree |

...

### Author(s)

Luke J. Harmon

### Examples

```
data(geospiza)
attach(geospiza)

name.check(geospiza.tree, geospiza.data)
```

---

## `node.leaves` *Returns all descendents of a given node in your tree*

---

### Description

Given a node number (as stored in the phylo object's edge matrix), this function returns a list of all descendents of that node.

### Usage

```
node.leaves(phy, node)
```

### Arguments

| | |
|---|---|
| `phy` | An object of class phylo |
| `node` | Node number |

### Value

A list of tip names

### Author(s)

Luke J. Harmon

### Examples

```
data(geospiza)
attach(geospiza)

node.leaves(geospiza.tree, 18)
```

---

## phy.anova *Phylogenetic ANOVA and MANOVA*

---

### Description

Calculates ANOVA or MANOVA, and returns p-value based on Brownian motion simulations

### Usage

```
phy.anova(phy, data, group, data.names=NULL, nsim=1000)
phy.manova(phy, data, group, data.names=NULL, nsim=1000, test="Wilks")
```

### Arguments

| | |
|---|---|
| `phy` | an object of class 'phylo' |
| `data` | Dependent variable(s) - should be continuous |
| `group` | Independent variable - should be a factor |
| `data.names` | Taxon names in order, corresponding to x and group; not needed if data has names stored as 'names' or 'rownames' |
| `nsim` | Number of simulations for calculating p-value; default = 1000 |
| `test` | If you're using a MANOVA, the name of the test statistic to be used. Partial matching is used so the name can be abbreviated. Options are "Pillai", "Wilks", "Hotelling-Lawley", and "Roy" |

### Details

This function allows an ANOVA or MANOVA in a phylogenetic context. First, the test statistic for ANOVA (one dependent variable) or MANOVA (more than one dependent variable) is calculated. The null distribution of this test statistic is then obtained by simulating new sets of dependent variables on the phylogenetic tree. Simulations are run under a Brownian motion model. For ANOVA, the rate parameter is estimated from the average squared independent contrast; for MANOVA the simulations use an estimated variance-covariance (vcv) matrix from the GEIGER function ic.sigma.

For MANOVA, you can specify the test statistic for the summary table. Wilks' statistic is most popular in the literature; for more details see the summary.manova help page.

### Value

Standard ANOVA or MANOVA table and p-value based on simulations

### Author(s)

Luke J. Harmon

### References

Garland Jr., T., A. W. Dickerman, C. M. Janis, and J. A. Jones. 1993. Phylogenetic analysis of covariance by computer simulation. Syst. Biol. 42(3):265-292.

### See Also

anova, summary.manova, ic.sigma

### Examples

```
data(geospiza)
attach(geospiza)


f<-as.factor(c(rep(0, 7), rep(1, 6)))


phy.manova(geospiza.tree, geospiza.data, f)

x<-geospiza.data[,1]

phy.anova(geospiza.tree, x, f, data.names=rownames(geospiza.data))
```

---

## `prune.extinct.taxa` *Prune specified taxa from a phylogenetic tree*

---

### Description

Prunes a set of taxa from a tree, either a random fraction of all taxa, or just the taxa whos tip branch terminates before the present day (extinct taxa)

### Usage

```
prune.extinct.taxa(phy, tol = .Machine$double.eps^0.5)
prune.random.taxa(phy, n)
```

### Arguments

| | |
|---|---|
| `phy` | Phylogenetic tree |
| `tol` | Tolerance for taxa that do not reach the present day exactly. This prevents taxa from being pruned just due to rounding error |
| `n` | Number of random taxa to prune from the tree |

### Value

New tree without these taxa

### Author(s)

Luke J. Harmon

## Examples

```
# Birth-death tree with extinct taxa
p2<-birthdeath.tree(b=0.2, d=0.1, time.stop=30)
plot(p2)

# Previous tree with extinct taxa removed
prune.extinct.taxa(p2)->p3
if(!is.null(p3)) plot(p3)

p4<-birthdeath.tree(b=0.2, d=0, taxa.stop=100)
p5<-prune.random.taxa(p4, 40)

plot(p5)
```

---

## `rate.estimate` *Calculate net diversification rate with confindence limits, and test diversities*

---

### Description

rate.estimate Uses Magellon and Sanderson method to calculate net diversification rate for a clade given extant diversity and age. Associated functions crown.p and stem.p also calculate the probability of obtaining a clade with at least k species given a net diversification rate (r), extinction fraction (e), and time interval. Associated functions stem.limits and crown.limits generate confidence limits on extant diversity given a net diversification rate (r), extinction fraction (e), and time interval.

### Usage

```
rate.estimate(time=0, n=0, phy=NULL, epsilon = 0, missing = 0, crown=TRUE, kenda
crown.p(time, r, epsilon, n)
stem.p(time, r, epsilon, n)
crown.limits(r, epsilon, time, prob=c(0.025, 0.975))
stem.limits(r, epsilon, time, prob=c(0.025, 0.975))
```

### Arguments

| | |
|---|---|
| `time` | Time interval; can be a vector |
| `n` | Number of extant species |
| `phy` | Phylogenetic tree; can be supplied instead of time and n. If you're using a tree then crown is automatically true. |
| `epsilon` | Extinction rate as a fraction of speciation rate |
| `missing` | Number of taxa missing from tree |
| `crown` | If true, time is treated as crown age; otherwise, stem age |
| `kendall.moran` | If true, calculates Kendall-Moran estimate of speciation rate; requires a complete phylogenetic tree |
| `r` | Net diversification rate, birth - death |
| `prob` | Range of probabilities for calculating confidence region |

## Value

rate.estimate: Returns net diversification rate r = lambda - mu, and confidence interval crown.p and stem.p: Returns the probability of obtaining a clade as big as, or bigger than, size n, given time, r, and epsilon stem.limits and crown.limits: Return confidence intervals for clade size given time, r, and epsilon

## Author(s)

Luke J. Harmon and Chad Brock

## References

Magallon, S. and M. J. Sanderson. 2000. Absolute diversification rates in angiosperm clades. Evolution 55:1762-1780.

## Examples

```
data(geospiza)
attach(geospiza)

# Assuming no extinction
rate.estimate(phy=geospiza.tree, missing=1)

# Assuming high extinction
rate.estimate(phy=geospiza.tree, epsilon=0.9, missing=1)
```

---

`rc` *Relative cladogenesis test*

---

## Description

Carries out relative cladogenesis test for all slices through the tree.

## Usage

```
rc(phy, make.plot=TRUE, plot.bonf=FALSE, p.cutoff=0.05, cex=par("cex"))
```

## Arguments

| | |
|---|---|
| `phy` | Phylogenetic tree |
| `make.plot` | Plot tree with significant branches highlighted? |
| `plot.bonf` | Use bonferroni correction when highlighting significant branches on the tree plot? (see below for more details) |
| `p.cutoff` | Cutoff for significant p-values; only used when plotting |
| `cex` | A numeric value giving the factor scaling of the tip and node labels (Character EXpansion). The default is to take the current value from the graphical parameters. |

## Details

A list of nodes is returned, along with the number of lineages alive just before that node, the maximum number of descendents that any of those lineages has at the present day, a p-value for this observation under the null hypothesis of a birth-death process (that is, given the null, what is the probability that one of these lineages had at least that many descendents), and the p-value after Bonferroni correction (given that a total of n-1 comparisons are made).

If a plot is made, asterisks will mark significantly diverse clades. These asterisks appear just to the right of the MRCA of the diverse clade.

The Bonferroni correction used here is exceedingly conservative for a tree of any reasonable size. I'm not sure that I recommend it, especially given the exploratory nature of this test and the non-independence of the comparisons.

One will often see significant results "trickle down" nodes in the tree - that is, if one clade is expecially diverse, then one or more of its parent clades will also be diverse. I think the most parsimonious place to attribute this effect is to the most shallow significant branch - that is, the branch closest to the tips (see Moore et al. 2004).

## Value

Table of results with four columns: Number of ancestors, Maximum descendents, p-value, Bonferroni-corrected p-value

## Author(s)

Luke J. Harmon

## References

Purvis, A., Nee, S. & Harvey, P. H. (1995) Proc. R. Soc. London Ser. B 260, pp. 329-333. Moore, B.R., K.M.A. Chan, and M.J. Donoghue. 2004. Detecting diversification rate variation in supertrees. In O.R.P. Bininda-Emonds (ed.), Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, pp. 487-533. Kluwer Academic, Dordrecht, the Netherlands.

## Examples

```
data(geospiza)
attach(geospiza)

rc(geospiza.tree)
```

---

`runMedusa` *MEDUSA: Modeling evolutionary diversification using stepwise AIC*

---

## Description

Fits a series of diversification models to phylogenetic and taxonomic data using stepwise AIC

## Usage

```
runMedusa(phy, richness, estimateExtinction=T, modelLimit=20, cutAtStem=T, start
summaryMedusa(phy, richness, out, cutoff=4, plotTree=T, useCorrection=F, cutAtSt
```

## Arguments

| | |
|---|---|
| `phy` | Phylogenetic tree in phylo format |
| `richness` | A table with taxon names in column 1 and extant diversities in column 2 |
| `estimateExtinction` | If false, fits a series of pure-birth models; otherwise birth-death models are used |
| `modelLimit` | Maximum number of rate shifts to place on the tree |
| `cutAtStem` | When rate shifts are placed on branches in Medusa, they could be placed at either the beginning or the end of the stem branch. cutAtStem=T cuts at the beginning of the stem branch, while cutAtStem=F cuts at the end. |
| `...` | Additional arguments for nlm |
| `out` | Output from medusa, sent to summaryMedusa to compile results |
| `cutoff` | Cutoff value for differences in AIC scores when comparing models. More complex models with an AIC score more than this number of units lower than simpler models are retained. |
| `plotTree` | Plot phylogenetic tree with branches under different models shown in different colors. |
| `useCorrection` | If true, use AICc instead of AIC |
| `startR` | Starting guess for net diversification rate. Try tweaking if Medusa crashes |
| `startE` | Starting guess for relative extinction rate. Try tweaking if Medusa crashes |

## Details

MEDUSA fits a series of diversification models to a combination of phylogenetic and taxonomic data. The input is a phylogenetic tree with branch lengths proportional to time showing the relationship among clades, and the diversity for living species in all of those clades. All taxa missing from the tree have to be assigned to one of the tip clades in the richness matrix.

The algorithm first fits a Single diversification model to the entire dataset. Then, a series of breaks are added, so that different parts of the tree evolve with different parameter values (birth and or death rates). MEDUSA first compares all single-breakpoint models with the overall model, and selects the best one. Then all possible two-breakpoint models are compared with the best single-breakpoint model, and so on.

summaryMedusa summarizes the results of MEDUSA. Here one must choose a cutoff value for improvement in AIC score necessary to accept more complex models.

## Value

From medusa: a matrix where each row summarizes the next more complex model selected by the stepwise procedure. Columns record: the node where the next break was found, the likelihood, the number of parameters, the AIC score, and the AICc score (corrected for small sample size).

From summaryMedusa: a list summarizing the model selected using AIC (or AICc) scores and the cutoff value. Each item in the list gives the likelihood for that part of the tree, two parameter estimates, r=b-d and epsilon=d/b, and the tip taxa included in that part of the tree.

## Author(s)

Luke J. Harmon and Daniel Rabosky

### References

Alfaro, M. E., F. Santini, C. Brock, H. Alamillo, A. Dornburg, D. L. Rabosky, G. Carnevale, and L. J. Harmon. 2009. Nine exceptional radiations plus high turnover explain species diversity in jawed vertebrates. Proceedings of the National Academy of Sciences 106:13410-13414.

### Examples

```
data(geospiza)
attach(geospiza)
```

---

## `sim.char` *Simulate character evolution*

---

### Description

Simulates evolution of discrete or continuous characters on a phylogenetic tree

### Usage

```
sim.char(phy, model.matrix, nsims = 1, model = "brownian", root.state = 1)
```

### Arguments

| | |
|---|---|
| `phy` | Phylogenetic tree |
| `model.matrix` | Matrix describing model: either vcv matrix or q matrix |
| `nsims` | Number of simulations to run |
| `model` | "brownian", "speciational", or "discrete" |
| `root.state` | Starting state at root, only needed for discrete model. |

### Details

This function simulates either discrete or continuous data on a phylogenetic tree. The model variable determines the type of simulation to be run. There are three options: "discrete," which evolves characters under a continuous time Markov model, and two continuous models, "brownian" and "speciational." Brownian is a constant rate Brownian motion model, while speciational is a Brownian model on a tree where all branches have the same length. The model.matrix parameter gives the structure of the model, and should be either a transition matrix, Q, for model = "discrete," or a trait variance-covariance matrix for model = "brownian" or "speciational." For discrete models, one character at a time is simulated, and the size of the model.matrix gives the number of character states; for continuous models, multivariate characters can be simulated, with their evolution goverened by a covariance matrix specified in the model.matrix. "root.state" is only needed for the discrete model.

### Value

Array of simulated data, either two or three-dimensional. The first dimension is the number of taxa, the second the number of characters, and the third the number of simulated data sets.

### Author(s)

Luke J, Harmon

### Examples

```
data(geospiza)
attach(geospiza)


s<-ic.sigma(geospiza.tree, geospiza.data)
sims<-sim.char(geospiza.tree, s, 100)

# Discrete character evolution
q<-list(rbind(c(-.5, .5), c(.5, -.5)))
sims<-sim.char(geospiza.tree, q, model="discrete", n=100)
```

---

## `tip.disparity` *Tip disparity calculation*

---

### Description

Calculates disparity for every clade in a tree

### Usage

```
tip.disparity(phy, data, data.names=NULL, disp = "avg.sq")
```

### Arguments

| | |
|---|---|
| `phy` | object of type 'phylo' |
| `data` | Data matrix - either actual or simulated |
| `data.names` | Tip names for data vector that match tree species; ignored if data includes names |
| `disp` | Disparity metric |

### Value

Vector of disparities

### Author(s)

Luke J. Harmon

### References

Harmon et al. 2003

## Examples

```
data(geospiza)
attach(geospiza)

tip.disparity(geospiza.tree, geospiza.data)
```

---

## 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) will 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=500
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)->va
vmat(phy)->vm

## verify that both packages return identical results
print(all(va==vm))
```

# Index