Package 'NELSI'

March 24, 2014

Type Package

Title NELSI: Nucleotide EvoLution Simulator

Version 0.2

Date 2014-03-22

Author David Duchene and Sebastian Duchene

Maintainer Sebastian Duchene <sebastian.duchene@sydney.edu.au>

Description NELSI simulates rates of molecular evolution along phylogenetic trees.

Depends R (>= 2.15.0), ape (>= 3.0-8), epibase (>= 0.1-2), geiger (>= 1.99-3)

Suggests phangorn (>= 1.7-4)

License GPL (>=2)

R topics documented:

NELSI-package *NELSI: Nucleotide EvoLution Simulator*

Description

NELSI simulates rates of molecular evolution along phylogenetic trees.

Details

Package:	NELSI
Type:	Package
Version:	0.2
Date:	2014-03-22
License:	GPL (>= 2)

Author(s)

David Duchene and Sebastian Duchene Maintainer: Sebastian Duchene <sebastian.duchene@sydney.edu.au>

References

Pending.

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

# Simulate uncorrelated rates with default parameters:
rateTree.default <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm)
plot(rateTree.default, col.lineages = rainbow(50))
```

allnode.times *allnode.times*

Description

allnode.times is used to obtain the ages of all nodes in the tree. It is similar to branching.times() in ape, but it also returns the ages of the tips. This is particularly useful for heterochronous trees.

Usage

```
allnode.times(phylo, tipsonly = FALSE)
```

Arguments

phylo	A phylogenetic tree of class 'phylo'
tipsonly	A logical (T / F). With T, the function returns the ages of the tips only. With F, it returns the ages of the tips and internal nodes.

Details

This function is similar to branching.times, but it also returns the ages of the tips. In ultrametric trees, the ages of all tips are 0, but in heterocrhonous trees they can have different ages.

Value

A vector with the ages of all nodes in the tree. The youngest tip always has age 0. The items of the vector are numbered according to the 'phylo' object. To inspect the numbering of the tips and internal nodes see the example.

Note

None

Author(s)

David Duchene and Sebastian Duchene

References

Pending.

See Also

branching.times() from package 'ape'

Examples

```
set.seed(12345)
myTree <- rtree(10)
plot(myTree)
# See the numbering of internal nodes and tips. Note that the tip labels and the actual numbering of the ti
nodelabels()
tiplabels()
allnode.times(myTree)

# Plot the tree and add the ages of the tips and internal nodes.
plot(myTree, show.tip.label = FALSE)
allTimes <- allnode.times(myTree)
allTimes
tiplabels(round(allTimes[1:10]))
nodelabels(round(allTimes[11:19]))


## The function is currently defined as
function (phylo, tipsonly = FALSE)
{
    require(phangorn)
    di.phylo <- dist.nodes(phylo)
```

```
    root.phylo <- phylo$edge[, 1][!(phylo$edge[, 1] %in% phylo$edge[,
        2])][1]
    phylo.depth <- max(di.phylo[as.numeric(colnames(di.phylo)) ==
        root.phylo, ])
    node.times <- phylo.depth - di.phylo[as.numeric(rownames(di.phylo)) ==
        root.phylo, ]
    if (tipsonly == TRUE) {
        node.times <- phylo.depth - di.phylo[as.numeric(rownames(di.phylo)) ==
            root.phylo, 1:length(phylo$tip.label)]
    }
    return(node.times)
}
```

`get.clock.data` *get.clock.data*

Description

get.clock.data returns a data frame with the root-to-tip distance in units of time and substitutions. These data are useful to test for clock-like behaviour with a OLS regression. The function also prints a plot of substitutions vs. time. The results are similar to those produced in path-o-gen (2009)

Usage

```
get.clock.data(rate.sim.object, tipsonly = T, ...)
```

Arguments

`rate.sim.object`	An object of class ratesim, which is typically obtained with the rate simulation functions (see example).
`tipsonly`	Logical. root-to-tip regressions are usually conducted for the tips only, but it can also include the internal nodes. If tipsonly == T, the data only include the tips, if tipsonly == F it will include the tips and internal nodes.
`...`	Additional arguments passed to plot()

Details

None.

Value

A data frame with two colums. If tipsonly == T, the first colum is the time from the root to the tips and the second column is the number of substitutions from the root to the tip. If tipsonly == F, the data frame will also include the root-to-tip distances for the internal nodes.

Note

This function can also be used to analyse trees from a BEAST analysis.

Author(s)

Sebastian Duchene

References

For a the original method implemented in java see: Rambaut, A. "Path-O-Gen: temporal signal investigation tool." (2009). http://tree.bio.ed.ac.uk/software/pathogen/

See Also

trann2trdat can obtain the tree data matrix from a tree estimated with BEAST

Examples

```
set.seed(12345)
myTree <- rtree(10) # Note that this tree is not ultrametric.
myTreeTimes <- allnode.times(myTree)


plot(myTree, show.tip.label = FALSE)
tiplabels(round(myTreeTimes[1:10], 2))
nodelabels(round(myTreeTimes[11:19], 2))

# Simulate rates along the tree with the uncorrlated lognormal model with default settings.
rateTree <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm)

clockDataTree <- get.clock.data(rateTree, pch = 20, col = "blue")

# Linear regression for substitutions as a function of time

lmRate <- lm(substitutions ~ times, data = clockDataTree)
summary(lmRate)


## The function is currently defined as
function (rate.sim.object, tipsonly = T, ...)
{
    phylogram <- rate.sim.object$phylogram
    chrono <- rate.sim.object$phylogram
    chrono$edge.length <- rate.sim.object[[2]][, 7]
    times <- allnode.times(chrono, tipsonly)
    substitutions <- allnode.times(phylogram, tipsonly)
    plot(times, substitutions, ...)
    return(data.frame(times, substitutions))
  }
```

---

get.lineage.time.rate *get.lineage.time.rate*

Description

get.lineage.time.rate obtains the branch-wise rates along a lineage. It uses a taxon to trace the rates back to the root of the tree. It is used internally by plot.ratesim().

Usage

```
get.lineage.time.rate(taxon, sim.rate.object)
```

Arguments

taxon	An integer corresponding to the index of a taxon. The index is the order of the taxa in the tree, which can be obtained with tr$tip.label. See example.
sim.rate.object	An object of class ratesim. Usually produced with simulate.rate

Details

None

Value

A data frame with two colums. The rows correspond to the each of the branches from the taxon to the root of the tree. The first column of the data frame is the mid age of the branch, and the second column is the rate along the branch.

Note

None

Author(s)

Sebastian Duchene

References

None.

See Also

simulate.rate()

Examples

```
set.seed(123425)
myTree <- rcoal(10)
rateTree <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm)
plot(rateTree, col.lineages = rainbow(10))

# Get the rate for the lineages ascending from the first taxon
# Find the name of the first taxon
myTree$tip.label

get.lineage.time.rate(taxon = 1, sim.rate.object = rateTree)



## The function is currently defined as
function (taxon, sim.rate.object)
{
    tree.data.matrix <- sim.rate.object[[2]]
    chrono <- sim.rate.object[[1]]
    chrono$edge.length <- tree.data.matrix[, 7]
    taxon.init <- taxon
    if (taxon %in% tree.data.matrix[, 3]) {
```

```
        data.matrix <- tree.data.matrix
        branch.times <- vector()
        rate.time <- vector()
        repeat {
            parent <- data.matrix[, 2][data.matrix[, 3] == taxon]
            time.br <- data.matrix[, 4][data.matrix[, 3] == taxon]
            rate.br <- data.matrix[, 5][data.matrix[, 3] == taxon]
            rate.time <- c(rate.time, rate.br)
            branch.times <- c(branch.times, time.br)
            taxon <- parent
            if (!(parent %in% data.matrix[, 3])) {
                break
            }
        }
        first.rate <- rate.time[length(rate.time)]
        last.rate <- rate.time[1]
        rate.time <- c(last.rate, rate.time, first.rate)
        node.times <- allnode.times(chrono)
        root.age <- max(node.times)
        branch.times <- c(node.times[taxon.init], branch.times,
            root.age)
        return(data.frame(branch.times, rate.time))
    }
    else {
        stop("The taxon name was not found in the tree data matrix. It should be a number between 1 and the
    }
}
```

`get.rate.descendant.pairs` *get.rate.descendant.pairs*

Description

The functions returns a data.frame with the rates for all pairs of parent-daughter lineage pairs. This is useful to assess the covariance of rates, which can be used as a measure of rate autcorrelation.

Usage

```
get.rate.descendant.pairs(rate.sim.object)
```

Arguments

`rate.sim.object` An object of class ratesim, typically obtained with simulate.rate

Details

None

Value

A data.frame. Each row is a daughter-parent pair. The columns are the rate of the parent branch, the rate of the daughter branch, and their difference in mid branch lengths.

Note

None.

Author(s)

Sebastian Duchene

References

None

See Also

plot.ratesim ratesim

Examples

```
set.seed(1234525)
myTree <- rcoal(50)

#Simulate rates with no autocorrelation (uncorrelated rates)

rateTreeUncor <- simulate.rate(myTree, FUN = simulate.uncor.lnorm)

uncorRates <- get.rate.descendant.pairs(rateTreeUncor)

#Simulate rates with high autocorrelation

rateTreeAutocor <- simulate.rate(myTree, FUN = simulate.autocor.kishino, params = list(initial.rate = 0.01

autocorRates <- get.rate.descendant.pairs(rateTreeAutocor)

# Plot the rates through time for all lineages to inspect the degree of autocorrelation

plot(rateTreeAutocor, col.lineages = rainbow(50))
plot(rateTreeUncor, col.lineages = rainbow(50))

# Estimate correlation of branch and daughter branch-wise rates. The correlation coefficient should be high

cor(x = autocorRates$parent.rate, y = autocorRates$daughter.rate)

cor(x = uncorRates$parent.rate, y = uncorRates$daughter.rate)


## The function is currently defined as
function (rate.sim.object)
{
    dat <- rate.sim.object$tree.data.matrix
    parent.rate <- vector()
    daughter.rate <- vector()
    diff.br.len <- vector()
    for (i in 1:nrow(dat)) {
        daughter.temp <- dat[i, 3]
        parent.temp <- dat[i, 2]
        br.temp <- dat[i, 4]
```

```
        if (parent.temp %in% dat[, 3]) {
            parent.rate <- c(parent.rate, dat[dat[, 3] == parent.temp,
                5])
            daughter.rate <- c(daughter.rate, dat[i, 5])
            diff.br.len <- c(diff.br.len, abs(br.temp - dat[dat[,
                3] == parent.temp, 4]))
        }
    }
    return(data.frame(parent.rate, daughter.rate, diff.br.len))
}
```

`get.tree.data.matrix` *get.tree.data.matrix*

Description

Obtain the tree data matrix from a phylogenetic tree. The tree should be a chronogram, with branch lengths representing time.

Usage

```
get.tree.data.matrix(phylo)
```

Arguments

`phylo`	A phylogenetic tree of class 'phylo'. The tree should be a chronogram, with branch lengths representing time.

Details

None

Value

A matrix with 7 colums and number of rows = number of edges in the tree. Each row corresponds to a branch (or edge) of the tree. The columns of the matrix correspond to the following:

branch.index: The number of the branch (or edge) in the tree. parent.node: The node of the corresponding branch that is closer to the root of the tree. daughter.node: The node of the corresponding branch that is closer to the tips. branch.midage: The median age of the branch. branch.rate: The rate along the branch. It is set to NA if get.tree.data.matrix is called directly on a phylogenetic trees. The functions simulate.rate and trann2trdat fill this column with the branch-wise rates. lengths.subst: Number of substitutions along the branch. It is set to NA if get.tree.data.matrix is called directly on a phylogenetic trees. The functions simulate.rate and trann2trdat fill this column with the branch-wise rates. length.time: The branch lengths in units of time.

Note

This function is used internally by the simulate.rate()]

Author(s)

David Duchene and Sebastian Duchene

References

None.

See Also

simulate.rate

Examples

```
set.seed(12345)
myTree <- rcoal(10)
myDataMatrix <- get.tree.data.matrix(myTree)
print(myDataMatrix)


## The function is currently defined as
function (phylo)
{
    require(phangorn)
    require(geiger)
    data.matrix <- matrix(data = NA, ncol = 7, nrow = length(phylo$edge.length))
    colnames(data.matrix) <- c("branch.index", "parent.node",
        "daughter.node", "branch.midage", "branch.rate", "length.subst",
        "length.time")
    data.matrix[, 1] <- 1:length(phylo$edge.length)
    data.matrix[, 2] <- phylo$edge[, 1]
    data.matrix[, 3] <- phylo$edge[, 2]
    data.matrix[, 4] <- mid.edge.ages(phylo)
    data.matrix[, 7] <- phylo$edge.length
    class(data.matrix) <- "tree.data.matrix"
    return(data.matrix)
  }
```

`mid.edge.ages` *mid.edge.ages obtans the ages of the mid point of the branches of a phylogenetic tree. It is not necessary for the tree to be ultrametric.*

Description

mid.edge.ages produces a vector of length = number of edges of the tree with the mid.age of the branchies. Assuming that the tree is a chronogram.

Usage

```
mid.edge.ages(phylo)
```

Arguments

`phylo`	A phylogenetic tree of class 'phylo'. The branch lengths should be units of time.

Details

None

Value

A vector of length = number of edges in the tree. Each value is the median branch age.

Note

None

Author(s)

David Duchene

References

None

See Also

allnode.times produces the ages of internal nodes and tips.

Examples

```
set.seed(12345)
myTree <- rcoal(10)

plot(myTree)
midAges <- mid.edge.ages(myTree)
edgelabels(round(midAges, 2))

## The function is currently defined as
function (phylo)
{
    require(phangorn)
    rootage <- max(allnode.times(phylo))
    if (is.ultrametric(phylo) == TRUE) {
        midages <- vector()
        for (i in 1:length(phylo$edge.length)) {
            if (phylo$edge[i, 2] > length(phylo$tip.label)) {
                recent.node.age <- branching.times(phylo)[(phylo$edge[i,
                  2] - length(phylo$tip.label))]
                halflength <- phylo$edge.length[i]/2
                midages[i] <- recent.node.age + halflength
            }
            else {
                midages[i] <- phylo$edge.length[i]/2
            }
        }
        return(midages)
    }
    else {
        nodetimes <- vector()
        extantedgelen <- max(phylo$edge.length[as.vector(which(phylo$edge[,
            1] == as.numeric(names(which(branching.times(phylo) ==
            min(branching.times(phylo)))))))])
        addedval <- abs(min(branching.times(phylo))) + extantedgelen
        for (i in 1:length(branching.times(phylo))) {
            nodetimes[i] <- (rootage/(max(branching.times(phylo)) +
```

```
            addedval)) * (branching.times(phylo) + addedval)[i]
        }
        brlen <- vector()
        for (i in 1:length(phylo$edge.length)) {
            brlen[i] <- (rootage/(max(branching.times(phylo)) +
                addedval)) * phylo$edge.length[i]
        }
        midages <- vector()
        for (i in 1:length(brlen)) {
            if (phylo$edge[i, 2] > length(phylo$tip.label)) {
                daughter.node.age <- nodetimes[(phylo$edge[i,
                  2] - length(phylo$tip.label))]
                halflength <- brlen[i]/2
                midages[i] <- daughter.node.age + halflength
            }
            else {
                parent.node.age <- nodetimes[(phylo$edge[i, 1] -
                  length(phylo$tip.label))]
                midages[i] <- parent.node.age - (brlen[i]/2)
            }
        }
        return(round(midages, 5))
    }
}
```

---

pathnode *pathnode*

Description

pathnode obtains the root-to tip distance for all tips in the tree, and the number of nodes along the path to each tip. These data are useful to evaluate the Node Density effect in phylogenetic trees.

Usage

```
pathnode(phylo, tipsonly = T)
```

Arguments

phylo	A phylogenetic tree of class 'phylo'
tipsonly	A logical value (T / F) indicating whether the function should only return the root-to-tip distance and number of nodes along a lineage for the tips only. Select T for the tips only, and F for the tips and internal nodes.

Details

The function plots the root-to-tip distance and the number of nodes along each lineage. It also returns a list with the two variables. The data can be used to assess the node-density effect in phylogenetic trees.

Value

A list with two items:

roottotippath	A vector with the distance from the root to the tips. If tipsonly==F, it will also include the distances for internal nodes.
nodesinpath	A vector with the number of nodes along a lineage. If tipsonly==F, it will also include the number of nodes for internal nodes. These are counted from the tips to the root.

Note

Please see the references for more detailed information.

Author(s)

David Duchene

References

Venditti, Chris, Andrew Meade, and Mark Pagel. "Detecting the node-density artifact in phylogeny reconstruction." Systematic biology 55.4 (2006): 637-643.

Hugall, Andrew F., and Michael SY Lee. "The likelihood node density effect and consequences for evolutionary studies of molecular rates." Evolution 61.10 (2007): 2293-2307.

See Also

Pending.

Examples

```
set.seed(12345)
myTree <- rtree(10)

par(mfrow = c(1, 2))
plot(myTree)
nde <- pathnode(myTree)
nde



## The function is currently defined as
function (phylo, tipsonly = T)
{
    require(phangorn)
    di.tr <- dist.nodes(phylo)
    root.tr <- phylo$edge[, 1][!(phylo$edge[, 1] %in% phylo$edge[,
        2])][1]
    tr.depth <- max(di.tr[as.numeric(colnames(di.tr)) == root.tr,
        ])
    if (tipsonly == TRUE) {
        roottotippath <- di.tr[as.numeric(rownames(di.tr)) ==
            root.tr, 1:length(phylo$tip.label)]
        nodesinpath <- sapply(1:length(phylo$tip.label), function(x) length(Ancestors(phylo,
            x)))
    }
```

```
    else {
        roottotippath <- di.tr[as.numeric(rownames(di.tr)) ==
            root.tr, ]
nodesinpath <- sapply(1:(length(phylo$tip.label)+phylo$Nnode), function(x) length(Ancestors(phylo, x)))
    }
    plot(roottotippath, nodesinpath, xlab = "Root-to-tip path length",
        ylab = "Number of parent nodes", pch = 20)
    return(list(roottotippath = roottotippath, nodesinpath = nodesinpath))
  }
```

`plot.ratesim` *plot.ratesim*

Description

This function plots the rate through time for all lineages in the tree of a ratesim object in two pannels in the active graphics device. The first plot is the branch-wise rate vs the mid-age of the branch (rate vs. age). This is done for the branches that ascend from every tip in the tree. The second plot is the chronogram. The tips are coloured according to the lineages as shown in the rate vs. age plot. The width of the branches in the chronogram are proportional to their rate.

Usage

```
plot.ratesim(rate.sim.object, col.lineages = colors(), type = "l")
```

Arguments

`rate.sim.object`	An object of class ratesim, obtained with simulate.rate()
`col.lineages`	A vector with the colours for the lineages in the tree. The length should be the same as the number of tips in the tree. The default uses the first colours from the colors() function.
`type`	The type of plot. The default is "l", but "s" or "p" can represent the rates more accurately because in most models the rates are averages over the branches.

Details

none.

Value

The funtion plots the rate in the active graphics device.

Note

None.

Author(s)

Sebastian Duchene

References

None.

See Also

get.lineage.time.rate()

Examples

```
set.seed(123425)
myTree <- rcoal(10)
rateTree <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm)
plot.ratesim(rate.sim.object = rateTree, col.lineages = rainbow(10), type = "s")

# for a non-ultrametric tree
set.seed(123425)
myTree1 <- rtree(10)
rateTree1 <- simulate.rate(tree = myTree1, FUN = simulate.uncor.lnorm)
plot.ratesim(rate.sim.object = rateTree1, col.lineages = rainbow(10), type = "s")



## The function is currently defined as
function (rate.sim.object, col.lineages = colors(), type = "l")
{
    rates.time.list <- list()
    for (i in 1:length(rate.sim.object[[1]]$tip.label)) {
        rates.time.list[[i]] <- get.lineage.time.rate(i, rate.sim.object)
    }
    ylims <- range(lapply(rates.time.list, function(y) range(y[,
        2])))
    chrono <- rate.sim.object[[1]]
    chrono$edge.length <- rate.sim.object[[2]][, 7]
    node.ages <- allnode.times(chrono)
    xlims <- sort(range(node.ages), decreasing = T)
    par(mfrow = c(1, 2))
    plot(rates.time.list[[1]][, 1], rates.time.list[[1]][, 2],
        ylim = ylims, xlim = xlims, ylab = "Rate", xlab = "Time",
        type = type, lwd = 3, col = col.lineages[1])
    for (k in 2:length(rates.time.list)) {
        lines(rates.time.list[[k]][, 1], rates.time.list[[k]][,
            2], ylim = ylims, xlim = xlims, col = col.lineages[k],
            lwd = 3, type = type)
    }
    plot(chrono, edge.width = 1 + log(rate.sim.object[[2]][,
        5]/min(rate.sim.object[[2]][, 5])), show.tip.label = F,
        root.edge = T)
    tiplabels(pch = 16, col = col.lineages, cex = 1.5)
  }
```

`simulate.autocor.kishino` *simulate.autocor.kishino*

Description

Simulate rates of evolution along a phylogenetic tree with the model reported in Kishino et al. (2001).

Usage

```
simulate.autocor.kishino(tree, params = list(initial.rate = 0.01, v = 0.3))
```

Arguments

`tree`	A phylogenetic tree of class phylo. The tree should be a chronogram, with branch lengths in time.
`params`	parameters for the autocorrelation function. This should be a list with two items:
`initial.rate`	The rate at the root of the tree
`v`	This is the nu parameter described in Kishino et al. (2001). A high value implies low autocorrelation, with high differences in the rates of parent and daughter branches. A low value of v implies high autocorrelation, with very similar rates between parent and daughter branches.

Details

See the original reference for further details.

Value

An object of class 'ratesim'. This is a list with two items:

`phylogram`	The phylogenetic tree with branch lengths in units of substitutions (phylogram)
`tree.data.matrix`	This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None

Author(s)

Sebastian Duchene. But see the reference for the original method.

References

Kishino, H., Thorne, J.L., and Bruno, W. J. "Performance of a divergence time estimation method under a probabilistic model of rate evolution." Molecular Biology and Evolution 18.3 (2001): 352-361.

See Also

simulate.rate() can call all the rate simulation functions internally.

Examples

```
set.seed(1234525)
myTree <- rcoal(20)

#Simulate high autocorrelation
kishinoRateTreeHigh <- simulate.autocor.kishino(myTree, params = list(initial.rate = 0.01, v = 0.001))
plot(kishinoRateTreeHigh, col.lineages = rainbow(20))

#Simulate low autocorrelation
kishinoRateTreeLow <- simulate.autocor.kishino(myTree, params = list(initial.rate = 0.01, v = 0.5))
plot(kishinoRateTreeLow, col.lineages = rainbow(20))

## The function is currently defined as
function (tree, params = list(initial.rate = 0.01, v = 0.3))
{
    require(phangorn)
    require(geiger)
    initial.rate <- params$initial.rate
    v = params$v
    data.matrix <- get.tree.data.matrix(tree)
    while (any(is.na(data.matrix[, 5])) | any(is.nan(data.matrix[,
        5]))) {
        data.matrix[1, 5] <- initial.rate
        for (i in 2:nrow(data.matrix)) {
            parent.node <- data.matrix[i, 2]
            preceeding.parent <- data.matrix[, 2][data.matrix[,
                3] == parent.node]
            preceeding.parent.brage <- data.matrix[, 4][data.matrix[,
                2] == preceeding.parent][1]
            preceeding.parent.brrate <- data.matrix[, 5][data.matrix[,
                2] == preceeding.parent][1]
            if (!(is.na(preceeding.parent.brrate)) & !(is.nan(preceeding.parent.brrate)) &
                (parent.node %in% data.matrix[, 3])) {
                data.matrix[i, 5] <- abs(rlnorm(1, mean = log(abs(preceeding.parent.brrate)),
                  sd = v * data.matrix[i - 1, 7]^0.5))
            }
            else if (!(parent.node %in% data.matrix[, 3])) {
                data.matrix[i, 5] <- abs(rlnorm(1, mean = log(abs(initial.rate)),
                  sd = sqrt(initial.rate)))
            }
        }
    }
    data.matrix[, 6] <- data.matrix[, 7] * data.matrix[, 5]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
}
```

simulate.autocor.thorne

simulate.autocor.thorne

Description

Simulate rates of evolution along a phylogenetic tree with the model reported in Thorne et al. (1998).

Usage

```
simulate.autocor.thorne(tree, params = list(initial.rate = 0.01, v = 0.3))
```

Arguments

tree	A phylogeneti tree of class 'phylo'. The tree should be a chronogram, with branch lengths in time.
params	parameters for the autocorrelation function. This should be a list with two items:
initial.rate	The rate at the root of the tree
v	This is the nu parameter described in Kishino et al. (2001). A high value implies low autocorrelation, with high differences in the rates of parent and daughter branches. A low value of v implies high autocorrelation, with very similar rates between parent and daughter branches.

Details

See the original reference for further details.

Value

An object of class 'ratesim'. This is a list with two items:

phylogram	The phylogenetic tree with branch lengths in units of substitutions (phylogram)
tree.data.matrix	This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None

Author(s)

Sebastian Duchene. See the reference for the method.

References

Thorne, J.L., Kishino,H., and Painter, I.S. "Estimating the rate of evolution of the rate of molecular evolution." Molecular Biology and Evolution 15.12 (1998): 1647-1657.

See Also

simulate.autocor.kishino for a similar model

Examples

```
set.seed(1234525)
myTree <- rcoal(20)

#Simulate high autocorrelation
thorneRateTreeHigh <- simulate.autocor.thorne(myTree, params = list(initial.rate = 0.01, v = 0.001))
plot(thorneRateTreeHigh, col.lineages = rainbow(20))

#Simulate low autocorrelation
thorneRateTreeLow <- simulate.autocor.thorne(myTree, params = list(initial.rate = 0.01, v = 0.5))
plot(thorneRateTreeLow, col.lineages = rainbow(20))



## The function is currently defined as
function (tree, params = list(initial.rate = 0.01, v = 0.3))
{
    require(phangorn)
    require(geiger)
    initial.rate <- params$initial.rate
    v = params$v
    data.matrix <- get.tree.data.matrix(tree)
    while (any(is.na(data.matrix[, 5])) | any(is.nan(data.matrix[,
        5]))) {
        data.matrix[1, 5] <- initial.rate
        for (i in 2:nrow(data.matrix)) {
            parent.node <- data.matrix[i, 2]
            preceeding.parent <- data.matrix[, 2][data.matrix[,
                3] == parent.node]
            preceeding.parent.brage <- data.matrix[, 4][data.matrix[,
                2] == preceeding.parent][1]
            preceeding.parent.brrate <- data.matrix[, 5][data.matrix[,
                2] == preceeding.parent][1]
            if (!(is.na(preceeding.parent.brrate)) & !(is.nan(preceeding.parent.brrate)) &
                (parent.node %in% data.matrix[, 3])) {
                data.matrix[i, 5] <- abs(rlnorm(1, mean = log(abs(preceeding.parent.brrate)),
                  sd = v * abs(data.matrix[i, 4] - preceeding.parent.brage)^0.5))
            }
            else if (!(parent.node %in% data.matrix[, 3])) {
                data.matrix[i, 5] <- abs(rlnorm(1, mean = log(abs(initial.rate)),
                  sd = sqrt(initial.rate)))
            }
        }
    }
    data.matrix[, 6] <- data.matrix[, 7] * data.matrix[, 5]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

simulate.clock *simulate.clock*

Description

simulate.clock simulates a constant rate of evolution along a phylogenetic tree

Usage

```
simulate.clock(tree, params = list(rate = 0.02, noise = 1e-04))
```

Arguments

tree	A phylogeneti tree of class 'phylo'. The tree should be a chronogram, with branch lengths in time.
params	parameters for the clock rate simulation function. This should be a list with two items:
rate	The rate for the tree
noise	The stochastic noise. Note that if this parameter is set very high compared to the rate, the model will be similar to an uncorrelated rates model

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

phylogram	The phylogenetic tree with branch lengths in units of substitutions (phylogram)
tree.data.matrix	This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None

Author(s)

Sebastian Duchene. See references for the original description of the molecular clock.

References

This is a constant molecular rate model. The original model can be found in: Zuckerkandl, Emile, and Linus Pauling. "Molecular disease, evolution and genetic heterogeneity." (1962): 189-225.

Zuckerkandl, Emile, and Linus Pauling. "Evolutionary divergence and convergence in proteins." Evolving genes and proteins 97 (1965): 97-166.

See Also

None.

Examples

```
set.seed(1234525)

myTree <- rcoal(10)

# A tree with low stochastic variation
rateClock <- simulate.clock(tree = myTree, params = list(rate = 0.01, noise = 0.00001))

#Note the scale in the y axis. Rate variation is very low
plot(rateClock, col.lineages = rainbow(10))


## The function is currently defined as
function (tree, params = list(rate = 0.02, noise = 1e-04))
{
    rate <- params$rate
    noise <- params$noise
    data.matrix <- get.tree.data.matrix(tree)
    branch.rates <- rep(rate, times = length(tree$edge.length))
    branch.rates <- abs(branch.rates + rnorm(length(tree$edge.length),
        mean = 0, sd = noise))
    data.matrix[, 5] <- branch.rates
    data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

`simulate.rate` *simulate.rate is the generic function to simulate rates along phylogenetic trees with any of the models.*

Description

simulate.rate simulates the rate with any of the models implemented in NELSI

Usage

```
simulate.rate(tree, FUN, ...)
```

Arguments

`tree`	A phylogenetic tree of class 'phylo'. The branch lengths should be in units of time (a chronogram)
`FUN`	This is any of the rate simulation functions (please see the help for each for details on the models): - simulate.autocor.kishino - simulate.autocor.thorne - simulate.clock - simulate.tdep.generic - simulate.tdep.ho - simulate.uncor.exp - simulate.uncor.lnorm - simulate.uncor.gamma - simulate.white.noise
`...`	This should be the parameters for the rate models. To specify this use: params = list(parameter1 , parameter2). See the help files for each rate simulation function for details on the parameters.

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

`phylogram` The phylogenetic tree with branch lengths in units of substitutions (phylogram)

`tree.data.matrix`
This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None.

Author(s)

David Duchene and Sebastian Duchene

References

See the original reference for NELSI: Pending.

See Also

- simulate.autocor.kishino - simulate.autocor.thorne - simulate.clock - simulate.tdep.generic - simulate.tdep.ho - simulate.uncor.exp - simulate.uncor.lnorm - simulate.uncor.gamma - simulate.white.noise

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

# Simulate uncorrelated rates with default parameters:
rateTree.default <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm)
plot(rateTree.default, col.lineages = rainbow(50))

# Simulate uncorrelated rates with custom parameters:
rateTree.custom <- simulate.rate(tree = myTree, FUN = simulate.uncor.lnorm, params = list(mean.log = -3.9,
plot(rateTree.custom, col.lineages = rainbow(50))


## The function is currently defined as
function (tree, FUN, ...)
{
    ratesim.object <- FUN(tree, ...)
    return(ratesim.object)
  }
```

simulate.tdep.ho *simulate.tdep.ho*

Description

This function simulates time-dependent molecular rates for branches in a chronogram using an vertically transformed exponential curve (Ho et al. 2005).

Usage

```
simulate.tdep.ho(tree, params = list(mu = 0.035, srate = 0.015, lambda = 0.1, noise = 0.001))
```

Arguments

tree	A phylogenetic tree of class 'phylo', with branch lengths in terms of time.
params	These are the three parameters describing the variation in rate through time. mu is the instantaneous mutation rate, observed near the present. srate is the long term observed substitutions rate. lambda is the rate of the rate decrease towards the past. noise is the standard deviation of a normal distribution from which noise for the rate is added.

Details

The number of substitutions for each branch is calculated by integrating the function given by the user within the limits of the age of a branch. Values of substitutions are added noise and used to determine the rate at each branch.

Note that this is function only uses one of the possible models for time dependence (see simulate.tdep.generic for a more general form).

Value

An object of class "ratesim". This is similar to a list; the first element is of class "phylo" with branch lengths in terms of substitutions. The second is a "tree.data.matrix" which can be used as a "data.frame". The tree.data.matrix contains the edge object of the phylogeny, the mid age of a branch, and branch lengths in terms of substitutions, time, and molecular rate.

Note

None

Author(s)

David Duchene and Sebastian Duchene

References

Ho, S. Y., Phillips, M. J., Cooper, A., & Drummond, A. J. (2005). Time dependency of molecular rate estimates and systematic overestimation of recent divergence times. Molecular biology and evolution, 22(7), 1561-1568.

See Also

simulate.tdep.generic simulate.autocor.kishino simulate.autocor.thorne simulate.uncor.exp simulate.uncor.gamma simulate.uncor.lnorm simulate.white.noise simulate.clock

Examples

```
set.seed(12345)
myTree <- rcoal(50)
plot(myTree); axisPhylo()
# Perhaps a lamda value of 4 is more appropriate to simulate a significant rate change through time in this
plot(function(x) 0.015 + (0.035 * exp(-4 * x)), xlim = c(0, max(branching.times(myTree))))
rate.simulation <- simulate.tdep.ho(myTree, params = list(mu = 0.035, srate = 0.015, lambda = 4, noise = 0.
plot(rate.simulation[[2]][,4], rate.simulation[[2]][,5], pch = 19, xlab = "Mid age of branch", ylab = "Mole
plot(rate.simulation[[1]])

## The function is currently defined as
function (tree, params = list(mu = 0.035, srate = 0.015, lambda = 0.1,
    noise = 0.001))
{
    require(phangorn)
    require(geiger)
    mu <- params$mu
    srate <- params$srate
    lambda <- params$lambda
    noise <- params$noise
    fun.rate <- function(x, m = mu, s = srate, lam = lambda) {
        if (any(x >= 0)) {
            return(s + (m * exp(-lam * x)))
        }
        else {
            stop("x is cannot be a negative number")
        }
    }
    data.matrix <- get.tree.data.matrix(tree)
    node.ages <- allnode.times(tree)
    b.times <- c(rep(0, length(tree$tip.label)), node.ages[(length(tree$tip.label) +
        1):length(node.ages)])
    names(b.times) <- 1:length(b.times)
    ratetemp <- vector()
    for (i in 1:length(tree$edge.length)) {
        parentage <- b.times[as.character(data.matrix[i, 2])]
        daughterage <- b.times[as.character(data.matrix[i, 3])]
        ratetemp[i] <- integrate(fun.rate, lower = daughterage,
            upper = parentage)$value/data.matrix[i, 7]
    }
    data.matrix[, 5] <- abs(ratetemp + rnorm(nrow(data.matrix),
        mean = 0, sd = noise))
    data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

simulate.uncor.exp *simulate.uncor.exp*

Description

Simulate the rate of evolution along a phylogenetic tree using the exponential model described in Drummond et al. (2006)

Usage

```
simulate.uncor.exp(tree, params = list(mean.exp = 0.001))
```

Arguments

tree	A phylogenetic tree of class phylo. The branch lengths should be in units of time (chronogram)
params	parameters for the autocorrelation function. This should be a list with one item:
mean.exp	This is the mean rate of the exponential distribution. Note that this is the same as the sd for this distribution

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

phylogram	The phylogenetic tree with branch lengths in units of substitutions (phylogram)
tree.data.matrix	This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None

Author(s)

Sebastian Duchene. See the reference for the description of the model.

References

Drummond, A.J., et al. "Relaxed phylogenetics and dating with confidence." PLoS biology 4.5 (2006): e88.

See Also

None

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

rateTree <- simulate.uncor.exp(tree = myTree, params = list(mean.exp = 0.01))
plot(rateTree, col.lineages = rainbow(50))

#See the histogram of the branch-wise rates
hist(rateTree$tree.data.matrix[, 5])

## The function is currently defined as
function (tree, params = list(mean.exp = 0.001))
{
    mean.exp <- params$mean.exp
    data.matrix <- get.tree.data.matrix(tree)
    branch.rates <- rexp(n = length(tree$edge.length), rate = 1/mean.exp)
    data.matrix[, 5] <- branch.rates
    data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

---

simulate.uncor.gamma *simulate.uncor.gamma*

Description

Simulate the rate of evolution along a phylogenetic tree using the gamma distributed rates model described in Drummond et al. (2006)

Usage

```
simulate.uncor.gamma(tree, params = list(shape = 98, rate = 4361))
```

Arguments

tree	A phylogenetic tree of class 'phylo'. The branch lengths should be in units of time (chronogram)
params	parameters for the uncorrelated rates function. This should be a list with two items:
shape	The shape of the gamma distribution. It follows the usual parametrisation of the gamma distribution
rate	The rate of the gamma distribution. Note that this is different from the substitution rate. Instead, this the the rate parameter typically used for the gamma distribution

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

`phylogram` The phylogenetic tree with branch lengths in units of substitutions (phylogram)

`tree.data.matrix`
This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

Notes

Author(s)

Sebastian Duchene. See the reference for the original description of the model.

References

Drummond, A.J., et al. "Relaxed phylogenetics and dating with confidence." PLoS biology 4.5 (2006): e88.

See Also

simulate.uncor.lnorm simulate.uncor.exp

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

rateTree <- simulate.uncor.gamma(tree = myTree, params = list(shape = 98, rate = 4361))
plot(rateTree, col.lineages = rainbow(50))

#See the histogram of the branch-wise rates
hist(rateTree$tree.data.matrix[, 5])

## The function is currently defined as
function (tree, params = list(shape = 98, rate = 4361))
{
    shape.gamma <- params$shape
    rate.gamma <- params$rate
    data.matrix <- get.tree.data.matrix(tree)
    branch.rates <- rgamma(n = length(tree$edge.length), shape = shape.gamma,
        rate = rate.gamma)
    data.matrix[, 5] <- branch.rates
    data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

simulate.uncor.lnorm *simulate.uncor.lnorm*

Description

Simulate the rate of evolution along a phylogenetic tree using the lognormal model described in Drummond et al. (2006)

Usage

```
simulate.uncor.lnorm(tree, params = list(mean.log = -3.9, sd.log = 0.1))
```

Arguments

tree	A phylogenetic tree of class 'phylo'. The branch lengths should be in units of time (chronogram)
params	parameters for the uncorrelated rates function. This should be a list with two items:
mean.log	This is the mean rate of the lognormal distribution. It should be in log scale
sd.log	The standard deviation of the lognormal distribution

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

phylogram	The phylogenetic tree with branch lengths in units of substitutions (phylogram)
tree.data.matrix	This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

None

Author(s)

Sebastian Duchene. See the reference for the description of the model.

References

Drummond, A.J., et al. "Relaxed phylogenetics and dating with confidence." PLoS biology 4.5 (2006): e88.

See Also

simulte.uncor.exp

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

rateTree <- simulate.uncor.lnorm(tree = myTree, params = list(mean.log = -3.9, sd.log = 0.5))
plot(rateTree, col.lineages = rainbow(50))

#See the histogram of the branch-wise rates
hist(rateTree$tree.data.matrix[, 5])


## The function is currently defined as
function (tree, params = list(mean.log = -3.9, sd.log = 0.1))
{
    mean.log <- params$mean.log
    sd.log <- params$sd.log
    data.matrix <- get.tree.data.matrix(tree)
    branch.rates <- rlnorm(n = length(tree$edge.length), meanlog = mean.log,
        sdlog = sd.log)
    data.matrix[, 5] <- branch.rates
    data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
    tree$edge.length <- data.matrix[, 6]
    res <- list(tree, data.matrix)
    names(res) <- c("phylogram", "tree.data.matrix")
    class(res) <- "ratesim"
    return(res)
  }
```

`simulate.white.noise` *simulate.white.noise*

Description

simulate.white.noise simulates the rate along a phylogeneti tree according to a white noise process. In particular, it samples from a lognormal distribution with a given mean and standard deviation. For each branch a rate is drawn from a distribution with the given mean. The standard deviation is divided by the branch length divided by the mean of the branch lengths. As a result, the rate of a long branch will be drawn from a distribution with a lower standard deviation than that for a short branch (i.e. the rate is more uncertain for shorter branches).

Usage

```
simulate.white.noise(tree, params = list(mean.log = -3.9, sd.log = 0.1))
```

Arguments

`tree`	A phylogenetic tree of class 'phylo'. The branch lengths should be in units of time (a chronogram).
`params`	A list with the parameters for the simulation function, corresponding to the following two items:
`mean.log`	The mean of the lognormal distribution. It should be in log scale

`sd.log` The standard deviation of the lognormal distribution. The actual value used for each branch will vary inversely with the branch length. See the description of the function

Details

None

Value

An object of class 'ratesim'. This is a list with two items:

`phylogram` The phylogenetic tree with branch lengths in units of substitutions (phylogram)

`tree.data.matrix`
This is a matrix with the number of substitutions, rates, and times along every branch in the tree. See get.tree.data.matrix for more details

Note

none

Author(s)

Sebastian Duchene and David Duchene

References

A similar model is described in: Lepage, Thomas, et al. "A general comparison of relaxed molecular clock models." Molecular biology and evolution 24.12 (2007): 2669-2680.

See Also

simulate.uncor.lnorm

Examples

```
set.seed(1234525)

myTree <- rcoal(50)

rateTree <- simulate.white.noise(tree = myTree, params = list(mean.log = -3.9, sd.log = 0.1))
plot(rateTree, col.lineages = rainbow(50))

#See the histogram of the branch-wise rates
hist(rateTree$tree.data.matrix[, 5])

## The function is currently defined as
function (tree, params = list(mean.log = -3.9, sd.log = 0.1))
{
    mean.log <- params$mean.log
    sd.log <- params$sd.log
    data.matrix <- get.tree.data.matrix(tree)
    branch.noise <- sd.log / (data.matrix[, 7]/mean(data.matrix[,
        7]))
```

```
  branch.rates <- sapply(branch.noise, function(x) rlnorm(1,
      mean.log, branch.noise))
  data.matrix[, 5] <- branch.rates
  data.matrix[, 6] <- data.matrix[, 5] * data.matrix[, 7]
  tree$edge.length <- data.matrix[, 6]
  res <- list(tree, tree.data.matrix = data.matrix)
  class(res) <- "ratesim"
  return(res)
}
```

`trann2trdat` *trann2trdat. extract annontations from a nexus tree.*

Description

Convert a tree with annotations from BEAST to a data frame with the annotations for the rate, the branch lengths in substitions and the branch lengths in time. The tree should be read with funtion read.annotated.nexus from the epibase package.

Usage

```
trann2trdat(tree)
```

Arguments

`tree`	An object of class 'phylo' with annotations as attributes of the object. Normally the tree can be the output of BEAST or MRBayes, and it would be loaded in R with the function read.annotated.nexus from the epibase package.

Details

To load the tree use read.annotated.nexus("BEAST_output.tree"). This reads the annotations for each branch.

Value

A data frame with the branch name, parent node, daughter node, mid age, rate, number of substitutions, and time.

Note

see read.annotated.nexus from package epibase

Author(s)

David Duchene

References

None

See Also

see read.annotated.nexus from package epibase

Examples

```
## Not run:
myAnnotatedTree <- read.annotated.nexus("annotated.tree")
annnotationData <- trann2trdat(myAnnotatedTree)
head(annotationData)

## End(Not run)
## The function is currently defined as
function (tree)
{
    require(epibase)
    tree$edge.length <- unlist(sapply(tree$annotations, function(x) {
        x$length
    }))[1:length(tree$edge.length)]
    rates <- unlist(sapply(tree$annotations, function(x) {
        x$rate_median
    }))
    if (is.ultrametric(tree) == TRUE) {
        midages <- mid.edge.ages(tree)
    }
    else {
        midages <- mid.edge.ages(tree, max(unlist(sapply(tree$annotations,
            function(x) {
                x$height_median
            }))))
    }
    timelen <- tree$edge.length
    subslen <- tree$edge.length * rates
    return(data.frame(branch = rownames(as.data.frame(tree$edge)),
        parent = tree$edge[, 1], daughter = tree$edge[, 2], midage = midages,
        rate = rates, blensubs = subslen, blentime = timelen))
  }
```

Index