Package: TreeSim (via r-universe)

September 6, 2024

Type Package

Title Simulating Phylogenetic Trees

Version 2.2

Date 2015-10-05

Author Tanja Stadler

Maintainer Tanja Stadler `<tanja.stadler@bsse.ethz.ch>`

Depends ape, laser, geiger

Description Simulation methods for phylogenetic trees where (i) all tips are sampled at one time point or (ii) tips are sampled sequentially through time. (i) For sampling at one time point, simulations are performed under a constant rate birth-death process, conditioned on having a fixed number of final tips (sim.bd.taxa()), or a fixed age (sim.bd.age()), or a fixed age and number of tips (sim.bd.taxa.age()). When conditioning on the number of final tips, the method allows for shifts in rates and mass extinction events during the birth-death process (sim.rateshift.taxa()). The function sim.bd.age() (and sim.rateshift.taxa() without extinction) allow the speciation rate to change in a density-dependent way. The LTT plots of the simulations can be displayed using LTT.plot(), LTT.plot.gen() and LTT.average.root(). TreeSim further samples trees with n final tips from a set of trees generated by the common sampling algorithm stopping when a fixed number m>>n of tips is first reached (sim.gsa.taxa()). This latter method is appropriate for m-tip trees generated under a big class of models (details in the sim.gsa.taxa() man page). For incomplete phylogeny, the missing speciation events can be added through simulations (corsim()). (ii) sim.rateshifts.taxa() is generalized to sim.bdsky.stt() for serially sampled trees, where the trees are conditioned on either the number of sampled tips or the age. Furthermore, for a multitype-branching process with sequential sampling, trees on a fixed number of tips can be simulated using sim.bdtypes.stt.taxa(). This function further allows to simulate under epidemiological models with an exposed class.

The function sim.genespeciestree() simulates coalescent gene
trees within birth-death species trees, and sim.genetree()
simulates coalescent gene trees.

License GPL-2

LazyLoad yes

Repository https://phylotastic.r-universe.dev

RemoteUrl https://github.com/tanja819/TreeSim

RemoteRef HEAD

RemoteSha 7138f7b466a75d8fbff96e884a07b43e87826056

Contents

TreeSim-package *TreeSim: Simulating Phylogenetic Trees*

Description

Simulation methods for phylogenetic trees where (i) all tips are sampled at one time point or (ii) tips are sampled sequentially through time. (i) For sampling at one time point, simulations are performed under a constant rate birth-death process, conditioned on having a fixed number of final tips (sim.bd.taxa()), or a fixed age (sim.bd.age()), or a fixed age and number of tips (sim.bd.taxa.age()). When conditioning on the number of final tips, the method allows for shifts in rates and mass extinction events during the birth-death process (sim.rateshift.taxa()). The function sim.bd.age() (and sim.rateshift.taxa() without extinction) allow the speciation rate to change in a density-dependent way. The LTT plots of the simulations can be displayed using LTT.plot(), LTT.plot.gen() and LTT.average.root(). TreeSim further samples trees with n final tips from a set of trees generated by the common sampling algorithm stopping when a fixed number m»n of tips is first reached (sim.gsa.taxa()). This latter method is appropriate for m-tip trees generated under a big class of models (details in the sim.gsa.taxa() man page). For incomplete phylogeny, the missing speciation events can be added through simulations (corsim()). (ii) sim.rateshifts.taxa() is generalized to sim.bdsky.stt() for serially sampled trees, where the trees are conditioned on either the number of sampled tips or the age. Furthermore, for a multitype-branching process with sequential sampling, trees on a fixed number of tips can be simulated using sim.bdtypes.stt.taxa(). This function further allows to simulate under epidemiological models with an exposed class. The function sim.genespeciestree() simulates coalescent gene trees within birth-death species trees, and sim.genetree() simulates coalescent gene trees.

Details

Package:	TreeSim
Type:	Package
Version:	2.2
Date:	2015-10-05
License:	GPL-2
LazyLoad:	yes

Author(s)

Tanja Stadler <http://www.bsse.ethz.ch/cEvo>

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

`ape` `geiger`

`corsim` *corsim: Simulating the missing speciation events in an incomplete phylogenies.*

Description

corsim simulates the missing speciation event in an incomplete phylogeny assuming a constant speciation and extinction rate. These rates can be estimated with the functions bd.shifts.optim (if random speciation events are missing) and bd.groups.optim (if only young speciation events are missing) provided in TreePar. corsim allows to specify an upper and lower bound for the times of the missing speciation events.

Usage

```
corsim(x,lambda,mu,missing,told=0,tyoung=0)
```

Arguments

`x`	Vector of the speciation times in the incomplete phylogeny (where time is measured such that 0 is the present and increasing going into the past).
`lambda`	Speciation rate.
`mu`	Extincion rate.
`missing`	Number of missing species (i.e. missing speciation events).
`told`	Upper bound for the time of missing speciation events. Default is 0 which means no upper bound.
`tyoung`	Lower bound for the time of missing speciation events. Default is 0 which means no lower bound. tyoung<told unless tyoung=told=0 is required.

Value

`x`	Vector of speciation times: input and simulated speciation times.

Author(s)

Tanja Stadler

References

N. Cusimano, T. Stadler, S. Renner. A new method for handling missing species in diversification analysis applicable to randomly or non-randomly sampled phylogenies. Syst. Biol., 61(5): 785-792, 2012.

See Also

sim.bd.age, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree

Examples

```
# Speciation times of a tree with five species:
x<-c(1,1.5,3,5)
# We simulate using the following parameters:
lambda<-2
mu<-1
tyoung<-0.5
told<-4.5
# We simulate 5 additional speciation times (i.e. five additional species):
missing<-5

# xcompleted is x plus 5 additional speciation events between 0.5 and 4.5 timesteps
# in the past. xcompleted corresponds to a 10-species tree:
xcompleted<-corsim(x,lambda,mu,missing,told,tyoung)
```

cuttree *cuttree: Cutting off the tree to prune recent branches.*

Description

cuttree takes as input a tree and a cuttime, and then prunes all lineages more recent than cuttime.

Usage

```
cuttree(tree,cuttime)
```

Arguments

`tree`	Phylogenetic tree.
`cuttime`	Time before present at which all descendent lineages are deleted. Value between 0 (the present i.e. nothing is deleted) and the age of the tree (i.e. the whole tree is deleted).

Value

`tree`	Tree where all branches more recent than cuttime are pruned from the input tree.

Author(s)

Tanja Stadler

Examples

```
n<-10
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-2
age<-2

##
# Simulating numbsim reconstructed trees with n sampled species under a
# birth-death process with speciation rate lambda, extinction rate mu,
# sampling probability frac, and time age since origin:

trees<-sim.bd.taxa.age(n, numbsim, lambda, mu, frac, age, mrca = FALSE)

treec<-cuttree(trees[[1]],0.3)

plot(trees[[1]])
plot(treec)
```

`fast.tree` *fast.tree: Fast version of sim.bd.taxa for a single tree, with Brownian character tracking*

Description

fast.tree replicates the functionality of sim.bd.taxa for a single tree (more specifically, the internal function sim2.bd.reverse.single). When traits is set to TRUE it will also simulate the evolution of a character over the tree with Brownian motion, returning its result for every node and tip. In addition sample points across the length of the tree can be given and fast.tree will return a time slice of the character values for the extant branches given at these time points.

Usage

```
fast.tree = function(n, lambda, mu, frac = 1, traits = FALSE, sigma = 1, sampleTimes = 0.5)
```

Arguments

`n`	Number of extant sampled tips.
`lambda`	Speciation rate.
`mu`	Extinction rate.
`frac`	For use with incomplete sampling (see sim.bd.taxa). The actual number of tips is n/frac.
`traits`	When greater than zero, enables character tracking and sampling for the specified number of independent traits on the same tree.
`sigma`	The standard deviation of the Brownian motion of the character over time.
`sampleTimes`	A vector of times at which to sample, expressed as a proportion of the overall tree length (in time).

Value

If traits = FALSE, a list with two values, the first containing the simulated tree with n/frac extant tips, and the second containing the tree length. This format is used for consistency with sim2.bd.reverse.single.

If traits = TRUE, a list with items:

`tree`	The simulated tree with n/frac extant tips.
`samples`	A traits x max.leaves x n.samples matrix of characters taken at sampleTimes (padded with zeros), where traits is the number of independent traits, max.leaves is the maximum number of leaves at a sampled time and n.samples is the length of vector sampleTimes.
`n.leaves`	The number of leaves extant at each of the sampled times.
`sampleTimes`	The vector of sampled times (no longer expressed as a proportion).

Author(s)

Nick Beeton

See Also

sim.bd.taxa

Examples

```
##
# Demonstration of the two tree techniques
#

# use microbenchmark library to compare run times for a single tree
# (note that improvement decreases for multiple trees using sim.bd.taxa due to overheads)
library(microbenchmark)

# generate a single 1000 extant leaf random tree using raw fast.tree function
microbenchmark(fast.tree(n = 1000, lambda = 1.5, mu = 1, frac = 1), times = 1)
# generate the same tree using the sim.bd.taxa framework (which uses fast.tree)
microbenchmark(sim.bd.taxa(n = 1000, numbsim = 1, lambda = 1.5, mu = 1, frac = 1, fast = TRUE), times = 1)
# now generate a tree using the original technique
microbenchmark(sim.bd.taxa(n = 1000, numbsim = 1, lambda = 1.5, mu = 1, frac = 1, fast = FALSE), times = 1)
```

---

getx *getx: Calculating the vector of speciation / transmission times and sampling times for a phylogenetic tree.*

Description

getx calculates the vector of branching (speciation / transmission) times and sampling times for a phylogenetic tree (which may have polytomies). This vector is the input for the TreePar methods.

Usage

```
getx(datatree,sersampling)
```

Arguments

`datatree` Phylogenetic tree.

`sersampling` Set sersampling = 0 if all tips are from one timepoint; 1 otherwise.

Value

`x` Vector of branching times where 0 is the present and time increasing into the past; If sersampling = 1: Vector of branching and tip sampling times. Second column indicates for each time if branching event (1) or tip (0).

Author(s)

Tanja Stadler

Examples

```
### tree with tips sampled at one timepoint
n<-10
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-1
trees<-sim.bd.taxa(n, numbsim, lambda, mu, frac,complete=FALSE,stochsampling=TRUE)
branching<-getx(trees[[1]])

### tree with tips sampled sequentially through time
set.seed(1)
n<-10
lambda <- c(2,1,2)
mu <- c(1,0.5,1.5)
sampprob <-c(0.5,0.5,0.5)
times<-c(0,1,2)
numbsim<-2
trees<-lapply(rep(n,numbsim),sim.bdsky.stt,lambdasky=lambda,deathsky=mu,
timesky=times,sampprobsky=sampprob,rho=0,timestop=0)
branchingserial<-getx(trees[[1]][[1]],sersampling=1)
```

LTT.plot *LTT.plot: Plots the lineages through time of a set of phylogenetic trees.*

Description

LTT.plot plots the lineages through time (LTT) for a set of phylogenetic trees in black (complete or reconstructed; with or without polytomies) together with the average LTT plot in red. The trees may be simulated using any function in TreeSim, or may be empirical trees. The method works for ultrametric and non-ultrametric trees which are binary or have polytomies. NOTE: you probably need to adapt the code such that the plot is pretty for your particular data (range of axes etc).

Usage

```
LTT.plot(trees,width,precalc,bound=10^(-12),timemax,nmax,avg)
```

Arguments

`trees`	List with one or two entries. First component: list of phylogenetic trees; second component: vector with time of origins (can be empty).
`width`	Width of lines in plot.
`precalc`	Default = 0. If = 1, then parse ’LTT.plot.gen(trees)’ instead of ’trees’ for the input variable ’trees’.

`bound` Determines the value by which leaf times may differ in an ultrametric tree. If two tips are further apart than ’bound’, they are considered as sequentially sampled tips.

`timemax` Time axis is drawn from present=0 to timemax years in the past.

`nmax` Axis with number of species is drawn from 1 to nmax.

`avg` Default=FALSE. If true then the average LTT plot of all individual LTT plots is drawn in red.

Author(s)

Tanja Stadler

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

LTT.plot.gen, sim.bd.taxa, sim.bd.age, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree

Examples

```
# Simulation of a tree of age 10 under the density-dependent model
numbsim<-3
age<-10
lambda<-0.3
mu<-0
K<-40
tree<- sim.bd.age(age,numbsim,lambda,mu,mrca=FALSE,complete=FALSE,K=K)
# Plot of tree
LTT.plot(c(list(tree),list(c(age,age,age))))
#
# Simulation of a tree with 10 tips under the constant rate birth-death model
numbsim<-3
n<-10
lambda<-0.3
mu<-0
tree<- sim.bd.taxa(10,numbsim,lambda,mu,complete=FALSE,stochsampling=TRUE)
# Plot of tree
ages<-c()
for (i in 1:length(tree)){
 ages<-c(ages,tree[[i]]$root.edge+max(getx(tree[[i]])))
}
LTT.plot(c(list(tree),list(ages)),avg=TRUE)
```

LTT.plot.gen — *LTT.plot.gen: Calculates the number of lineages through time for each input tree, as well as the average number of lineages over all trees.*

Description

LTT.plot.gen calculates the number of lineages through time for each input tree, as well as the average number of lineages over all trees. The trees may be simulated using any function in TreeSim, or may be empirical trees. The method works for ultrametric and non-ultrametric trees which are binary or have polytomies.

Usage

```
LTT.plot.gen(trees,bound=10^(-12))
```

Arguments

`trees`	List with one or two entries. First component: list of phylogenetic trees; second component: vector with time of origins (can be empty).
`bound`	Determines the value by which leaf times may differ in an ultrametric tree. If two tips are further apart, they are considered as sequentially sampled tips.

Value

`out`	out[[1]]: First column are the branching times of ALL input trees. Second column is the number of lineages after the branching time. out[[i]]: Equivalent vector as out [[1]], but for tree i-1.

Author(s)

Tanja Stadler

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

LTT.plot,sim.bd.taxa, sim.bd.age, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree

Examples

```
# Simulation of a tree with a mrca at time 10 in the past,
# under the density-dependent model
numbsim<-10
age<-10
lambda<-0.3
mu<-0.2
```

```
K<-40
# You can produce LTT plots as follows.
# (for now this is un-commented, as some combinations of geiger / TreeSim on certain platforms
# produce problems. If this is the case for you, please report to tanjs.stadler@bsse.ethz.ch).
```

---

sim.bd.age — *sim.bd.age: Simulating birth-death trees of a fixed age.*

Description

sim.bd.age simulates trees conditioned on (i) the time since origin or (ii) the time since the most recent common ancestor of the extant tips. The method allows for incomplete sampling: only a fixed fraction of all tips is included in the sampled tree. The method assumes constant birth and death rates, or allows for a density-dependent birth rate. If you want to have species-age dependent rates, use sim.age in R package TreeSimGM.

Usage

```
sim.bd.age(age, numbsim, lambda, mu, frac = 1, mrca = FALSE,
 complete = TRUE, K = 0)
```

Arguments

age	Time since origin / most recent common ancestor.
numbsim	Number of trees to simulate.
lambda	Speciation rate.
K	If K=0, then lambda is constant. If K>0, density-dependent speciation is assumed, with speciation rate = lambda(1-m/K) when there are m extant species.
mu	Extinction rate.
frac	Sampling fraction: The actual number of tips is n/frac, but only n tips are included (incomplete sampling).
mrca	If mrca=FALSE: age is the time since origin. If mrca=TRUE: age is the time since most recent common ancestor of the extant tips.
complete	If complete = TRUE, the tree with the extinct and non-sampled lineages is returned. If complete = FALSE, the extinct and non-sampled lineages are suppressed.

Value

treearray	Array of 'numbsim' trees with the time since origin / most recent common ancestor being 'age'. If tree goes extinct or no tips are sampled (only possible when mrca = FALSE), return value is '0'. If only one extant and no extinct tips are sampled, return value is '1'.

Author(s)

Tanja Stadler

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

sim.bd.taxa, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree, sim.age

Examples

```
age<-2
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-3

##
# Simulating trees with time age since the origin:

sim.bd.age(age,numbsim,lambda,mu,mrca=FALSE,complete=TRUE)
sim.bd.age(age,numbsim,lambda,mu,frac,mrca=FALSE,complete=FALSE)

# Simulating trees with time age since the time of the most recent common
# ancestor of the extant and sampled species:

sim.bd.age(age,numbsim,lambda,mu,mrca=TRUE,complete=TRUE)
sim.bd.age(age,numbsim,lambda,mu,frac,mrca=TRUE,complete=FALSE)
```

---

sim.bd.taxa — *sim.bd.taxa: Simulating birth-death trees on a fixed number of extant taxa.*

Description

sim.bd.taxa simulates trees on n species under the constant rate birth-death process. The method allows for incomplete sampling, i.e. (i) only a fixed fraction of all extant tips is included in the sampled tree or (ii) each extant tip from a complete tree is included with a fixed probability. In both cases, the tree is conditioned to have n tips after sampling. If you want to relax the assumption of constant rates, this function will not work. If you want to change rates through time use sim.rateshift.taxa. If you want to have species-age dependent rates, use sim.taxa in R package TreeSimGM.

Usage

```
sim.bd.taxa(n, numbsim, lambda, mu, frac = 1, complete = TRUE,
 stochsampling = FALSE, fast = TRUE)
```

Arguments

`n` Number of extant sampled tips.

`numbsim` Number of trees to simulate.

`lambda` Speciation rate.

`mu` Extinction rate.

`frac` When complete = FALSE and stochsampling=FALSE: The actual number of tips is n/frac, but only n tips are included (incomplete sampling). When complete = FALSE and stochsampling=TRUE: Each tip is included into the final tree with probability frac. When complete = TRUE: all extinct and non-sampled lineages are included, i.e. the tree has n/frac extant tips.

`complete` If TRUE, the tree with the extinct and non-sampled lineages is returned. If FALSE, the extinct lineages are suppressed.

`stochsampling` See frac.

`fast` Use a faster version of the simulation that takes advantage of compiled Fortran code. See fast.tree for additional functionality using this method.

Value

`out` List of numbsim simulated trees with n extant sampled tips.

Note

For stochsampling = TRUE: The algorithm is fast for the critical process, lambda=mu.

Author(s)

Tanja Stadler, Nick Beeton

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

T. Stadler. On incomplete sampling under birth-death models and connections to the sampling-based coalescent. Jour. Theo. Biol. 261: 58-66, 2009.

See Also

sim.bd.age, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree, sim.taxa, fast.tree

Examples

```
n<-10
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-2

##
```

```
# Simulating numbsim trees with n species under a birth-death process with
# speciation rate lambda an extinction rate mu:

sim.bd.taxa(n,numbsim,lambda,mu)

# Each extant species is included in final tree with probability frac
# (the tree has n species AFTER sampling):

sim.bd.taxa(n,numbsim,lambda,mu,frac,complete=FALSE,stochsampling=TRUE)

# A fraction frac of the extant species is included into the final tree
# (the tree has n species AFTER sampling):

sim.bd.taxa(n,numbsim,lambda,mu,frac,complete=FALSE,stochsampling=FALSE)

##
# Demonstration of the two tree techniques
#

# use microbenchmark library to compare run times for a single tree
# (note that improvement decreases for multiple trees using sim.bd.taxa due to overheads)
library(microbenchmark)

# generate a single 1000 extant leaf random tree using raw fast.tree function
microbenchmark(fast.tree(n = 1000, lambda = 1.5, mu = 1, frac = 1), times = 1)
# generate the same tree using the sim.bd.taxa framework (which uses fast.tree)
microbenchmark(sim.bd.taxa(n = 1000, numbsim = 1, lambda = 1.5, mu = 1, frac = 1, fast = TRUE), times = 1)
# now generate a tree using the original technique
microbenchmark(sim.bd.taxa(n = 1000, numbsim = 1, lambda = 1.5, mu = 1, frac = 1, fast = FALSE), times = 1)

##
# suite of tests to compare both tree generating techniques

# generate 10,000 random 100-leaf trees using fast and original techniques
# This will generally take a few minutes to run
N = 10000
test1 = sim.bd.taxa(100, N, 1.5, 1, fast = FALSE)
test2 = sim.bd.taxa(100, N, 1.5, 1, fast = TRUE)

# Examine number of nodes - histograms should look similar
nnode1 = unlist(lapply(test1, Nnode))
nnode2 = unlist(lapply(test2, Nnode))
hist(nnode1, breaks = 10*(10:80), col='#0000FF80', ylim = c(0,N/10), main = 'Number of nodes')
hist(nnode2, breaks = 10*(10:80), col='#FF000080', add = TRUE)

# Examine overall tree length
tl1 = unlist(lapply(test1, function(x) max(node.depth.edgelength(x))))
tl2 = unlist(lapply(test2, function(x) max(node.depth.edgelength(x))))
hist(tl1, breaks = 0.5*(0:60), col='#0000FF80', ylim = c(0,N/10), main = 'Tree length (time)')
hist(tl2, breaks = 0.5*(0:60), col='#FF000080', add = TRUE)

# Examine mean node depth
nd1 = unlist(lapply(test1, function(x) mean(node.depth(x))))
```

```
nd2 = unlist(lapply(test2, function(x) mean(node.depth(x))))
hist(nd1, breaks = 0.5*(0:60), col='#0000FF80', ylim = c(0,N/10), main = 'Mean node depth (time)')
hist(nd2, breaks = 0.5*(0:60), col='#FF000080', add = TRUE)
```

sim.bd.taxa.age *sim.bd.taxa.age: Simulating birth-death trees with a given age on a fixed number of extant taxa.*

Description

sim.bd.taxa.age simulates trees on n species with a (i) fixed time since origin or (ii) fixed time since the most recent common ancestor of the sampled species under the constant rate birth-death process. The method allows for incomplete sampling, i.e. each extant tip from a complete tree is included with a fixed probability. The tree is conditioned to have n tips after sampling and a fixed time since origin or since the most recent common ancestor of the sampled species.

Usage

```
sim.bd.taxa.age(n, numbsim, lambda, mu, frac = 1, age, mrca = FALSE)
```

Arguments

`n`	Number of extant sampled tips.
`numbsim`	Number of trees to simulate.
`lambda`	Speciation rate.
`mu`	Extinction rate.
`frac`	Each tip is included into the final tree with probability frac.
`age`	The time since origin / most recent common ancestor.
`mrca`	If mrca = FALSE: The time since the origin of the process. If mrca = TRUE: The time since the most recent common ancestor of the sampled species.

Value

`treearray`	Array of numbsim trees with n>1 tips with a given age. The extinct lineages are not included.

Note

The algorithm is fast for the critical process, lambda=mu.

Author(s)

Tanja Stadler

References

T. Stadler: On incomplete sampling under birth-death models and connections to the sampling-based coalescent. J. Theo. Biol. (2009) 261: 58-66.

See Also

sim.bd.age, sim.rateshift.taxa, sim.gsa.taxa, birthdeath.tree

Examples

```
n<-10
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-2
age<-2

##
# Simulating numbsim reconstructed trees with n sampled species under a
# birth-death process with speciation rate lambda, extinction rate mu,
# sampling probability frac, and time age since origin:

sim.bd.taxa.age(n, numbsim, lambda, mu, frac, age, mrca = FALSE)

# Simulating numbsim reconstructed trees with n sampled species under a
# birth-death process with speciation rate lambda, extinction rate mu,
# sampling probability frac, and time age since the most recent
# common ancestor of the extant sampled species:

sim.bd.taxa.age(n, numbsim, lambda, mu, frac, age, mrca = TRUE)
```

---

sim.bdsky.stt *sim.bdsky.stt: Simulating sequentially sampled birth-death, SIS, SIR or SIRS trees where birth and death rates are changing through time.*

Description

sim.bdsky.stt simulates birth-death trees with tips being sampled sequentially. The birth and death rates may change in a piecewise fashion. The birth rates may additionally depend on the number of susceptible individuals in an epidemic, corresponding to epidemiological SIS, SIR or SIRS dynamics. The trees are conditioned on a fixed number of tips or a fixed age.

Usage

```
sim.bdsky.stt(n,lambdasky,deathsky,timesky,sampprobsky,omegasky=rep(0,length(timesky)),
rho=0,timestop=0,model="BD",N=0,trackinfecteds=FALSE)
```

Arguments

`n` If n>0, then the simulation is stopped once 'n' tips are sampled sequentially through time. If n=0, the simulation is stopped after time 'timestop'.

`lambdasky` Vector of dimension k, where k is the number of different birth rates. An individual between time (timesky[i],timesky[i+1]) has birth rate lambdasky[i].

`deathsky` Vector of dimension k, where k is the number of different death rates. An individual between time (timesky[i],timesky[i+1]) has death rate deathsky[i].

`timesky` Vector of dimension k, containing the times of rate shifts. Time is measured forward in time (unlike the function sim.rateshift.taxa where shifts are measured backward in time), with the origin of the tree being at time 0, i.e. timesky[1]=0.

`sampprobsky` Vector of dimension k, an individual dying during time (timesky[i],timesky[i+1]) is sampled with probability sampprobsky[i], i.e. is being included into the final tree.

`omegasky` Leave to default unless SIRS model simulation is being performed. omegasky is a vector of dimension k, where k is the number of different loose immunity rates (i.e. the rates of R->S transition in SIRS model). An individual between time (timesky[i],timesky[i+1]) has loose immunity rate omegasky[i].

`rho` Default is rho=0. If rho>0 and timestop>0, then the process is stopped after timestop and each individual alive at time timestop is included into the final tree with probability rho.

`timestop` Default is timestop=0, meaning the simulation is stopped once n tips are sampled. If timestop>0, then the simulation is stopped after time timestop.

`N` Total population size is N. Set N>0 when simulating under either of SIS/SIR/SIRS models.

`model` Should be set to desired model. Default="BD" (birth-death skyline model). The paramter accepts values "BD","SIS","SIR" or "SIRS". For all the models but "BD" N>0 should be set.

`trackinfecteds` Set to TRUE records prevalence and incidence data, i.e. number of overall infecteds and times of infections in epidemiological terms, or overall number of species in that clade since time of origin, and times of speciations in macroevolutionary terms.

Value

`out` List containing the phylogenetic tree with n sampled tips or a fixed age timestop for trackinfecteds=FALSE. If trackinfecteds=TRUE, the list contains also a second item, a list that tracks numbers of susceptible/infected/recovered/sampled individuals over the course of the tree growth. This list consist of: $timesky - times of rate changes, $eventtimes - times dating events happening in the tree, i.e. bifurcation, death, sampling, or loose immunity, $infecteds - the number of infected infividuals at $eventtimes, $cumulativeinfecteds - the cumulative number of infected individuals at $eventtimes, $cumulativesampleds - the cumulative number of sampled individuals at $eventtimes, and in case of SIR/SIRS model, the list also contains: $susceptibles - the number of susceptible individuals at $eventtimes, and $recovereds - the number of recovered individuals at $eventtimes. Times $timesky and $eventtimes are stated backward-in-time, such that time=0 is the time of the most recent sample, and time is increasing into the past. This allows for determining precisely the times of rate changes for skyline tree analyses. The counts in $infecteds,$cumulativeinfecteds, $cumulativesampleds, $susceptibles and $recovereds represent the number of individuals in each category prior to (more ancestral than) the $eventtimes.

Note

A large number of trees can be obtained using the R function lapply. The tree can be plotted using the R package ape function plot(tree). sim.bdsky.stt function extends the function sim.rateshift.taxa to trees which contain tips being sampled sequentially.

Author(s)

Tanja Stadler, Veronika Boskova

References

T. Stadler, D. Kuehnert, S. Bonhoeffer, A. Drummond. Birth-death skyline plot reveals temporal changes of epidemic spread in HIV and hepatitis C virus (HCV). Proc. Nat. Acad. Sci., 110(1): 228-233, 2013.

V. Boskova, S. Bonhoeffer, T. Stadler. Inference of epidemiological dynamics based on simulated phylogenies using birth-death and coalescent models. Manuscript.

See Also

sim.bdtypes.stt.taxa

Examples

```
### Set the values for birth rates (lambda), deathrates (mu),
# sampling proportion (sampprob) and times of rate shifts (times).
# Also set the number of sampled tips in the final tree (n) and
# the number of simulations (numbsim).
set.seed(1)
n<-10
lambda <- c(2,1,2)
mu <- c(1,0.5,1.5)
sampprob <-c(0.5,0.5,0.5)
times<-c(0,1,2)
numbsim<-2
# Simulate trees under the birth-death skyline model
trees<-lapply(rep(n,numbsim),sim.bdsky.stt,lambdasky=lambda,deathsky=mu,
timesky=times,sampprobsky=sampprob,rho=0,timestop=0)

### Simulate 10 trees with 100 tips under the SIRS model with
# total population size N=500
trees<-lapply(rep(100,10),sim.bdsky.stt,lambdasky=c(3,0.5,3,0.5,3),
deathsky=c(0.5,0.5,0.5,0.5,0.5),sampprobsky=c(0.5,0.5,0.5,0.5,0.5),
timesky=c(0,1,2,3,4),trackinfecteds=TRUE,model="SIRS",N=500,
omegasky=c(0,0.5,0.5,0.5,0))

### Simulate 1 tree with 100 tips under the SIRS model with
# total population size N=500 and plot the S,I,R classes
trees<-sim.bdsky.stt(100,lambdasky=c(3,0.5,3,0.5,3),deathsky=c(0.5,0.5,0.5,0.5,0.5),
sampprobsky=c(0.5,0.5,0.5,0.5,0.5),timesky=c(0,2,2.5,3,3.2),trackinfecteds=TRUE,
model="SIRS",N=500,omegasky=c(0,0.5,0.5,0.5,0.5))
plot(trees[[2]]$eventtimes,trees[[2]]$infecteds,xlim=rev(range(trees[[2]]$eventtimes)),
type="l",col="red",ylim=c(min(trees[[2]]$recovereds,trees[[2]]$infecteds,trees[[2]]$susceptibles),
max(trees[[2]]$recovereds,trees[[2]]$infecteds,trees[[2]]$susceptibles)),
xlab="time",ylab="Number of individuals")
abline(v=trees[[2]]$timesky,lty=2)
points(trees[[2]]$eventtimes,trees[[2]]$recovereds,type="l",col="green")
points(trees[[2]]$eventtimes,trees[[2]]$susceptibles,type="l",col="blue")
points(trees[[2]]$eventtimes,trees[[2]]$cumulativesampleds,type="l",col="grey")
legend("topleft",c("S","I","R","samples","rate changes"),
col=c("blue","red","green","grey","black"),lty=c(1,1,1,1,2))
```

sim.bdtypes.stt.taxa *sim.bdtypes.stt.taxa: Simulating multitype birth-death trees with a fixed number of tips sampled through time.*

Description

sim.bdtypes.stt.taxa simulates trees on n tips sampled through time under a multitype birth-death process.

Usage

```
sim.bdtypes.stt.taxa(n,lambdavector,deathvector,
sampprobvector,init=-1,EI=FALSE,eliminate=0)
```

Arguments

`n` Number of sampled tips.

`lambdavector` Matrix of dimension kxk, where k is the number of different states. The entry (i,j) is the rate with which an individual in state i gives rise to a new lineage of state j.

`deathvector` Vector of dimension k, the entry i is the death rate of an individual in state i.

`sampprobvector` Vector of dimension k, the entry i is the probability of an individual in state i being sampled upon death, i.e. being included into the final tree.

`init` Default is -1, meaning the initial individual is in a random state (which is chosen from the equilibrium distribution of states). If init>0, then the initial state is 'init'.

`EI` If EI=TRUE a model with two types, namely exposed and infectious individuals, is assumed. Infectious individuals transmit and give rise to exposed individuals with rate lambdavector[2,1], and exposed individuals become infectious with rate lambdavector[1,2]. Exposed individuals have a 0 death rate and cannot be sampled. For an example simulation see below.

`eliminate` Only relevant if EI=TRUE. Under EI=TRUE all sampled tips are in state 2. If eliminate>0, the first eliminate tips are marked with state 1. This facilitates further analysis, e.g. we now can easily prune these first eliminate tips to mimic no sampling close to the epidemic outbreak.

Value

`out` Phylogenetic tree with 'n' sampled tips. In out$states, the states for the tips are stored.

Note

A large number of trees can be obtained using the R function lapply. The tree can be plotted using the R package ape function plot(out). sim.bdtypes.stt.taxa function extends the simulator in the R package diversitree to trees which contain tips being sampled sequentially.

Author(s)

Tanja Stadler

References

T. Stadler, S. Bonhoeffer. Uncovering epidemiological dynamics in heterogeneous host populations using phylogenetic methods. Phil. Trans. Roy. Soc. B, 368 (1614): 20120198, 2013.

See Also

sim.bdsky.stt

Examples

```
# Simulate two trees with 10 tips
set.seed(1)
n<-10
lambda <- rbind(c(2,1),c(3,4))
mu <- c(1,1)
sampprob <-c(0.5,0.5)
numbsim<-2
trees<-lapply(rep(n,numbsim),sim.bdtypes.stt.taxa,
lambdavector=lambda,deathvector=mu,sampprobvector=sampprob)

# Testing the model with exposed class (EI = TRUE)
set.seed(2)
# simulate tree with expected incubation period of 14 days,
# infectious period of 7 days, and R0 of 1.5:
mu <- c(0,1/7)
```

```
lambda <- rbind(c(0,1/14),c(1.5/7,0))
# sampling probability of infectious individuals is 0.35:
sampprob <-c(0,0.35)
# we stop once we have 20 samples:
n <- 20
# we simulate one tree:
numbsim<-1
# We mark first eliminate=10 tips such that we can easily drop them later
# (if deleting these 10 tips, we mimic no sampling close to the outbreak)
trees<-lapply(rep(n,numbsim),sim.bdtypes.stt.taxa,lambdavector=lambda,deathvector=mu,
sampprobvector=sampprob,EI=TRUE,eliminate=10)
```

sim.genespeciestree *sim.genespeciestree: Simulating birth-death species trees with nested coalescent gene trees.*

Description

sim.genespeciestree simulates birth-death species trees (using sim.bd.taxa or sim.bd.taxa.age). Within each species tree, a gene tree is simulated, assuming a coalescent with coalescent rate being 1. The method returns summary statistics for the gene tree.

Usage

```
sim.genespeciestree(n, numbsim, lambda, mu, frac = 1, age=0)
```

Arguments

`n` Number of extant sampled tips.

`numbsim` Number of trees to simulate.

`lambda` Speciation rate.

`mu` Extinction rate.

`frac` Each tip is included into the final species tree with probability frac.

`age` The time since origin / most recent common ancestor. If age = 0 (default) a uniform prior for the time since origin is assumed.

Value

`statistics` For each simulated gene tree the following statistics are returned (with "gammaspecies" being the gamma statistic for the corresponding species tree): "Colless","s","Sackin","cherries","matching of species tree and gene tree","gammaspecies","gamma".

Author(s)

Tanja Stadler

References

T. Stadler, J. Degnan, N. Rosenberg. Manuscript.

Examples

```
#Simulate two gene trees within two species trees:
n<-10
lambda <- 2.0
mu <- 0.5
frac <-0.6
numbsim<-2
age<-2

# Simulation is conditioned on 10 final tips
sim.genespeciestree(n, numbsim, lambda, mu, frac, 0)

# Simulation is conditioned on 10 final tips and tree age 2
sim.genespeciestree(n, numbsim, lambda, mu, frac, age)
```

---

sim.genetree *sim.genetree: Simulating coalescent gene trees.*

Description

sim.genetree simulates a gene tree assuming the coalescent with coalescent rate being 1. The method returns summary statistics of the gene tree.

Usage

```
sim.genetree(n, numbsim)
```

Arguments

`n`	Number of extant sampled tips.
`numbsim`	Number of trees to simulate.

Value

`statistics`	For each simulated gene tree the following statistics are returned: "Colless","s","Sackin","cherries".

Author(s)

Tanja Stadler

References

T. Stadler, J. Degnan, N. Rosenberg. Manuscript.

Examples

```
n<-10
numbsim<-2

sim.genetree(n, numbsim)
```

---

sim.gsa.taxa *sim.gsa.taxa: Sampling trees on n tips from bigger trees.*

Description

sim.gsa.taxa samples trees on n tips (using the GSA approach, see references) from trees with m tips where m>n, given the m-tip trees are simulated under the simple sampling approach (i.e. simulating until first m»n tips are reached or the tree is extinct). The TreeSim methods to simulate n-tip trees, sim.bd.taxa and sim.rateshift.taxa, are implemented such that sim.gsa.taxa is not necessary. sim.gsa.taxa is needed for post processing of trees generated NOT in TreeSim: if the aim is to simulate trees with n co-existing tips, then typically simulators stop once the first time n co-existing lineages are reached. However, due to death, we can observe n tips later (e.g. n+1 lineages followed by death leads n lineages). sim.gsa.taxa produces an appropriate set of n-tip trees where the input are m-tip trees with m»n and the m-tip trees are simulated under these typical simulators.

sim.gsa.taxa works for m-tip trees generated under a model where: (i) the number of tips eventually tends to zero or stays bigger than n and (ii) birth / death rate changes do not depend on the time between the change and the present - e.g. one cannot model a mass extinction event 1 million years BEFORE the present. But one can model a mass extinction event 1 million years AFTER the origin of the tree. The package TreeSimGM uses sim.gsa.taxa to obtain n-tip trees.

Usage

```
sim.gsa.taxa(treearray, n, frac = 1, sampling = 1, complete = TRUE)
```

Arguments

`treearray`	Array of trees with a fixed number of tips.
`n`	Number of tips in sampled trees.
`frac`	Relevant when complete = FALSE: The actual number of tips is n/frac, but only n tips are included (incomplete sampling). When complete = TRUE: We set frac = 1.
`sampling`	Parameter determining how close the returned trees in treearray are to the "true" distribution. The higher ’sampling’, the closer the output trees to the ’true’ distribution. Default is 40. Higher values of sampling return fewer output trees meaning a larger input treearray is needed.
`complete`	If TRUE, the tree with the extinct lineages is returned. If FALSE, the extinct lineages are suppressed.

Value

`treearray`	Array of sampled trees with n extant sampled tips. Note that the number of trees in the output is significantly smaller than the number of trees in the input (in order to ensure correct tree sampling).

Author(s)

Tanja Stadler

References

K. Hartmann, D. Wong, T. Stadler. Sampling trees from evolutionary models. Syst. Biol., 59(4): 465-476, 2010.

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

sim.bd.age, sim.bd.taxa, sim.rateshift.taxa, birthdeath.tree

Examples

```
##
# First 100 trees on 9 tips under a birth-death process are generated.
# Then trees on 5 species are sampled from these 100 trees using the GSA
# (see references).
# You can easily simulate trees on m species with the simple sampling
# approach (see references) under a variety of models. Then use the
# provided GSA algorithm to get a correct sample of trees on n<<m species:

m<-9
n<-5
numbsim<-100
lambda <- 2.0
mu <- 0.5

t<-sim.bd.taxa(m,numbsim,lambda,mu)
t2<-sim.gsa.taxa(t,n)
```

`sim.rateshift.taxa` *sim.rateshift.taxa: Simulating trees incorporating mass extinction events and rate shifts.*

Description

sim.rateshift.taxa simulates trees on n species under the constant rate birth-death process. At user-specified points in the past, the rates can shift. Further, mass extinction events can be incorporated. The method further allows for incomplete sampling, i.e. only a fixed fraction of all tips is included in the sampled tree. The tree is conditioned to have n tips after sampling.

Usage

```
sim.rateshift.taxa(n, numbsim, lambda, mu, frac, times, complete = TRUE, K=0, norm = TRUE)
```

Arguments

`n` Number of extant sampled tips.

`numbsim` Number of trees to simulate.

`lambda` Vector of speciation rates, the rate in entry i is the speciation rate prior (ancestral) to time times[i].

`mu` Vector of extinction rates, the rate in entry i is the extinction rate prior (ancestral) to time times[i].

`frac` Vector of proportion of species surviving mass extinction event. Entry i corresponds to the mass extinction at time times[i]. If frac[i]=1, only rate shift but no mass extinction at time times[i].

`times` Vector of mass extinction and rate shift times. Time is 0 today and increasing going backwards in time. Specify the vector as times[i]<times[i+1]. times[1]=0 (today).

`complete` If TRUE, the tree including the extinct lineages and non-sampled lineages is returned (so the tree has round(n/frac[1]) extant tips). If FALSE, the extinct lineages and non-sampled lineages are suppressed.

`K` If K>0, then a density-dependent speciation rate = lambda*(1-numberspecies/K) is used. Only works currently for mu=0.

`norm` If norm = TRUE the simulations are exact. If norm = FALSE tree is always returned once N=0 in Stadler 2011, p.678, point (7).

Value

`out` List of numbsim simulated trees with n extant sampled tips.

Author(s)

Tanja Stadler

References

T. Stadler. Simulating trees on a fixed number of extant species. Syst. Biol., 60: 676-684, 2011.

See Also

sim.bd.age, sim.bd.taxa, sim.gsa.taxa, birthdeath.tree

Examples

```
n<-10
numbsim<-1

##
```

```
# Simulating trees with a fixed number of species having shifts in rate
# and mass extinction events.
# Between today and time 0.3 in the past, we have speciation rate 2,
# extinction rate 0. At time 0.3, we have a mass extinction event which
# 10% of the species survive. Prior to 0.3, we have a speciation rate
# of 1 and an extinction rate of 0.3:

sim.rateshift.taxa(n,numbsim,c(2,1),c(0,0.3),
c(1,0.1),c(0,0.3),complete=TRUE)

# The fraction 0.6 of the extant species is included into the final tree
# (the tree has n species AFTER sampling, extinct and
# non-sampled lineages are not included):

sim.rateshift.taxa(n,numbsim,c(2,1),c(0,0.3),
c(0.6,0.1),c(0,0.3),complete=FALSE)
```

Index