Package ‘coalescentMCMC’

March 3, 2015

Version 0.4-1

Date 2015-03-03

Title MCMC Algorithms for the Coalescent

Depends ape, coda, lattice

Imports Matrix, phangorn, stats

ZipData no

Description Flexible framework for coalescent analyses in R. It includes a main function running the MCMC algorithm, auxiliary functions for tree rearrangement, and some functions to compute population genetic parameters.

License GPL (>= 2)

Author Emmanuel Paradis [aut, cre, cph]

Maintainer Emmanuel Paradis <`Emmanuel.Paradis@ird.fr`>

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-03-03 22:37:48

R topics documented:

`coalescentMCMC` *Run MCMC for Coalescent Trees*

Description

This function runs a Markov chain Monte Carlo (MCMC) algorithm to generate a set of trees which is returned with their likelihoods.

`getMCMCtrees` extracts the trees from previous MCMC runs.

`saveMCMCtrees` saves the lists of trees from previous runs on the user's hard disk.

`cleanMCMCtrees` deletes the lists of trees from previous runs (the files possibly on the hard disk are not changed).

`getLastTree` extracts the last tree from a list of trees (object of class `"multiPhylo"`).

`getMCMCstats` returns the summary data for the different chains run during a session.

Usage

```
coalescentMCMC(x, ntrees = 3000, burnin = 1000, frequency = 1,
               tree0 = NULL, model = NULL, printevery = 100)
getMCMCtrees(chain = NULL)
saveMCMCtrees(destdir = ".", format = "RDS", ...)
cleanMCMCtrees()
getLastTree(X)
getMCMCstats()
```

Arguments

`x`	a set of DNA sequences, typically an object of class `"DNAbin"` or `"phyDat"`.
`ntrees`	the number of trees to output.
`burnin`	the number of trees to discard as "burn-in".
`frequency`	the frequency at which trees are sampled.
`tree0`	the initial tree of the chain; by default, a UPGMA tree with a JC69 distance is generated.
`model`	the coalescent model to be used for resampling. By default, a constant-THETA is used.
`printevery`	an integer specifying the frequency at which to print the numbers of trees proposed and accepted; set to 0 to cancel all printings.
`chain`	an integer giving which lists of trees to extract
`destdir`	a character string giving the location where to save the files; by default, this is the current working directory.
`format`	the format of the tree files. Three choices are possible (cae-insensitive): `"RDS"`, `"Newick"`, `"NEXUS"`, or any unambiguous abbreviation of these.
`...`	options passed to the function used to write the tree files (see below).
`X`	an bject of class `"multiPhylo"`.

Details

A simple MCMC algorithm is programmed using at each step the “neighborhood rearrangement” operation (Kuhner et al., 1995) and Hastings’s ratio for acceptance/rejection of the proposed tree.

The number of generations of the chain is determined by: ‘frequency’ times ‘ntrees’ plus ‘burnin’. Only the ‘ntrees’ trees are output whereas all the log-likelihood values are output.

The list of trees is returned in a specific environment and can be extracted with `getMCMCtrees`.

`saveMCMCtrees` saves the files with, by default, the RDS format using `saveRDS`. If `format = "Newick"`, `write.tree` is used.; if `format = "NEXUS"`, `write.nexus` is used. Options can be passed to any of these functions with `...`.

`getLastTree(X)` is a short-cut to `X[[length(X)]]`.

Value

`coalescentMCMC` returns an object of class `"coda"` with the log-likelihood and the parameters of each tree.

`getLastTree` returns an object of class `"phylo"`.

`getMCMCstats` returns a data frame.

Author(s)

Emmanuel Paradis

References

Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**, 97–109.

Kuhner, M. K., Yamato, J. and Felsenstein, J. (1995) Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. *Genetics*, **140**, 1421–1430.

See Also

`dcoal`, `treeOperators`

Examples

```
## Not run:
data(woodmouse)
out <- coalescentMCMC(woodmouse)
plot(out)
getMCMCtrees() # returns 3000 trees

## End(Not run)
```

`dcoal` *Density Functions of Various Time-Dependent Coalescent Models*

Description

These functions compute the (log-)likelihood values for various coalescent models, including the constant-Θ model and various time-dependent models.

Usage

```
dcoal(phy, theta, log = FALSE)
dcoal.step(phy, theta0, theta1, tau, log = FALSE)
dcoal.linear(phy, theta0, thetaT, TMRCA, log = FALSE)
dcoal.time(phy, theta0, rho, log = FALSE)
dcoal.time2(phy, theta0, rho1, rho2, tau, log = FALSE)
```

Arguments

`phy`	a tree of class `"phylo"`.
`theta`	population parameter THETA.
`log`	a logical value specifying whether the probabilities should be returned log-transformed.
`theta0, theta1, thetaT`	THETA parameter for the time-dependent models.
`tau`	breakpoint in time when the parameters change.
`TMRCA`	time to most recent ancestor.
`rho, rho1, rho2`	population (exponential) growth rates.

Details

The models are detailed in a vignette: `vignette("CoalescentModels")`.

Value

a numeric vector with (log-)likelihood values.

Author(s)

Emmanuel Paradis

References

Griffiths, R. C. and Tavaré, S. (1994) Sampling theory for neutral alleles in a varying environment. *Philosophical Transactions of the Royal Society of London. Series B. Biological Sciences*, **344**, 403–410.

Kuhner, M. K., Yamato, J. and Felsenstein, J. (1998) Maximum likelihood estimation of population growth rates based on the coalescent. *Genetics*, **149**, 429–434.

See Also

`coalescentMCMC`

`sim.coalescent` *Coalescent Simulation and Visualisation*

Description

This is pedagogic function to show what is the coalescent in a simple population model with discrete generations and asexual reproduction.

Usage

```
sim.coalescent(n = 5, TIME = 50, growth.rate = NULL, N.0 = 50, N.final = 20,
               col.lin = "grey", col.coal = "blue", pch = NULL, ...)
```

Arguments

`n`	the sample size.
`TIME`	the number of generations.
`growth.rate`	the growth rate of the population.
`N.0`	the initial size of the population.
`N.final`	the final size of the population (i.e., at present).
`col.lin`	the colour used to show links of ancestry in the population.
`col.coal`	the colour used to show the coalescent of the n individuals.
`pch`	the symbol used to show individuals (none by default).
`...`	further arguments passed to `points` if `pch` is used.

Details

The simulation works along the following steps. The number of individuals at each generation is calculated. For each individual, a (unique) parent is randomly chosen at the previous generation. All individuals are then plotted and the ancestry lines are shown; the individuals are eventually ordered to avoid line-crossings. A sample of n individuals are randomly chosen from the last generation, and their shared ancestry is shown with thicker lines.

The first (oldest) generation is at the bottom, and the final (present) one is at the top of the plot.

The population size at each generation is determined from the four arguments: `TIME`, `growth.rate`, `N.0`, and `N.final`. At least three of them must be given by the user. If `TIME` is not given, its value is calculated with `log(N.final/N.0) / growth.rate`.

This code was used to make the figures in Emerson et al. (2001).

Author(s)

Emmanuel Paradis

References

Emerson, B., Paradis, E. and Thebaud C. (2001). Revealing the demographic histories of species using DNA sequences. *Trends in Ecology and Evolution*, **16**, 707–716.

Examples

```
sim.coalescent()
sim.coalescent(N.0 = 20) # constant population size
```

`treeOperators` *Trees Operators for Running MCMC*

Description

These functions provide tools for tree rearrangement to be used as operators in a MCMC run.

Usage

```
NeighborhoodRearrangement(phy, n, nodeMax, target, THETA, brtimes)
TipInterchange(phy, n)
EdgeLengthJittering(phy)
```

Arguments

`phy`	a tree of class `"phylo"`.
`n`	the number of tips in `phy`.
`nodeMax`	the largest (integer) value of the node coding in `phy`.
`target`	the number of the node where the rearrangement will be done.
`THETA`	The estimate of Θ for `phy` at the node 'target'.
`brtimes`	the branching times of `phy`.

Details

`NeighborhoodRearrangement` performs a rearrangement as described by Kuhner et al. (1995).

`TipInterchange` interchanges two tips under the condition that they are not sisters.

`EdgeLengthJittering` alters the branch lengths by adding a random value from a uniform distribution defined by `range(phy$edge.length)` (the ultrametric nature of the tree is conserved).

Value

an object of class `"phylo"`.

Author(s)

Emmanuel Paradis

References

Kuhner, M. K., Yamato, J. and Felsenstein, J. (1995) Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. *Genetics*, **140**, 1421–1430.

See Also

`coalescentMCMC`, `dcoal`

Examples

```
tr <- rcoal(10)
ts <- NeighborhoodRearrangement(tr, 10, 19, 15, 1, branching.times(tr))
layout(matrix(1:2, 2))
plot(tr); plot(ts)
layout(1)
```

Index