Package: treedater (via r-universe)

October 30, 2024

Type Package

Title Fast Molecular Clock Dating of Phylogenetic Trees with Rate Variation

Version 0.5.3

Date 2022-03-10

Author Erik Volz [aut, cre]

Maintainer Erik Volz <erik.volz@gmail.com>

Description Functions for estimating times of common ancestry and molecular clock rates of evolution using a variety of evolutionary models, parametric and nonparametric bootstrap confidence intervals, methods for detecting outlier lineages, root-to-tip regression, and a statistical test for selecting molecular clock models. The methods are described in Volz, E.M. and S.D.W. Frost (2017) <doi:10.1093/ve/vex025>.

License GPL-2

Depends ape (>= 5.0),limSolve (>= 1.5.5.0)

Suggests lubridate,ggplot2,foreach,iterators,mgcv,knitr,rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.0

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

RemoteUrl https://github.com/emvolz/treedater

RemoteRef HEAD

RemoteSha 7b8a72aa0ea71ded4cde6cd287529aa8ae679c68

Contents

`treedater-package` *treedater fits a molecular clock to a phylogenetic tree and estimates evolutionary rates and times of common ancestry.*

Description

Additional functions are provided for detecting outlier lineages (possible sequencing or alignment error). A statistical test is available for choosing between strict and relaxed clock models. The calendar time of each sample must be specified (possibly with bounds of uncertainty) and the length of the sequences used to estimate the tree. treedater uses heuristic search to optimise the TMRCAs of a phylogeny and the substitution rate. An uncorrelated relaxed molecular clock accounts for rate variation between lineages of the phylogeny which is parameterised using a Gamma-Poisson mixture model.

Author(s)

Maintainer: Erik Volz <`erik.volz@gmail.com`>

References

Volz, E. M., and S. D. W. Frost. "Scalable relaxed clock phylogenetic dating." Virus Evolution 3.2 (2017).

`boot` *Estimate of confidence intervals of molecular clock parameters with user-supplied set of bootstrap trees*

Description

If the original treedater fit estimated the root position, root position will also be estimated for each simulation, so the returned trees may have different root positions. Some replicates may converge to a strict clock or a relaxed clock, so the parameter estimates in each replicate may not be directly comparable. It is possible to compute confidence intervals for the times of particular nodes or for estimated sample times by inspecting the output from each fitted treedater object, which is contained in the $trees attribute.

Usage

```
boot(
  td,
  tres,
  ncpu = 1,
  searchRoot = 1,
  overrideTempConstraint = TRUE,
  overrideClock = NULL,
  quiet = TRUE,
  normalApproxTMRCA = FALSE,
  parallel_foreach = FALSE
)
```

Arguments

`td`	A fitted treedater object
`tres`	A list or multiPhylo with bootstrap trees with branches in units of substitutions per site
`ncpu`	Number of threads to use for parallel computation. Recommended.
`searchRoot`	See *dater*
`overrideTempConstraint`	If TRUE (default) will not enforce positive branch lengths in simualtion replicates. Will speed up execution.
`overrideClock`	May be 'strict' or 'additive' or 'relaxed' in which case will force simulations to fit the corresponding model. If ommitted, will inherit the clock model from td
`quiet`	If TRUE will minimize output printed to screen
`normalApproxTMRCA`	If TRUE will use estimate standard deviation from simulation replicates and report confidence interval based on normal distribution
`parallel_foreach`	If TRUE will use the foreach package for parallelization. May work better on HPC systems.

Value

A list with elements

- trees: The fitted treedater objects corresponding to each simulation
- meanRates: Vector of estimated rates for each simulation
- meanRate_CI: Confidence interval for substitution rate
- coef_of_variation_CI: Confidence interval for rate variation
- timeOfMRCA_CI: Confidence interval for time of common ancestor

Author(s)

Erik M Volz <erik.volz@gmail.com>

See Also

dater parboot

Examples

```
# simulate a tree
tre <- ape::rtree(25)
# sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
# make a list of trees that simulate outcome of bootstrap using nonparametric phylogeny estimation
# also modify edge length to represent evolutionary distance with rate 1e-3:
bootTrees <- lapply( 1:25, function(i) {
 .tre <- tre
 .tre$edge.length <- tre$edge.length * pmax(rnorm( length(tre$edge.length), 1e-3, 1e-4 ), 0 )
 .tre
})
tre$edge.length <- tre$edge.length * 1e-3
# run treedater
td <- dater( tre, sts, s= 1000, clock='strict', omega0=.0015  )
# bootstrap:
( tdboot <- boot( td, bootTrees ) )
# plot lineages through time :
plot( tdboot )
```

`dater` *Estimate a time-scaled tree and fit a molecular clock*

Description

Estimate a time-scaled tree and fit a molecular clock

Usage

```
dater(
  tre,
  sts,
  s = 1000,
  omega0 = NA,
  minblen = NA,
  maxit = 100,
  abstol = 1e-04,
  searchRoot = 5,
  quiet = TRUE,
  temporalConstraints = TRUE,
  clock = c("strict", "uncorrelated", "additive"),
  estimateSampleTimes = NULL,
  estimateSampleTimes_densities = list(),
  numStartConditions = 1,
  clsSolver = c("limSolve", "mgcv"),
  meanRateLimits = NULL,
  ncpu = 1,
  parallel_foreach = FALSE
)
```

Arguments

`tre`	An ape::phylo which describes the phylogeny with branches in units of substitutions per site. This may be a rooted or unrooted tree. If unrooted, the root position will be estimated by checking multiple candidates chosen by root-to-tip regression. If the tree has multifurcations, these will be resolved and a binary tree will be returned.
`sts`	Vector of sample times for each tip in phylogenetic tree. Vector must be named with names corresponding to tre$tip.label.
`s`	Sequence length (numeric). This should correspond to sequence length used in phylogenetic analysis and will not necessarily be the same as genome length.
`omega0`	Vector providing initial guess or guesses of the mean substitution rate (substitutions per site per unit time). If not provided, will guess using root to tip regression.
`minblen`	Minimum branch length in calendar time. By default, this will be the range of sample times (max - min) divided by sample size.
`maxit`	Maximum number of iterations
`abstol`	Difference in log likelihood between successive iterations for convergence.
`searchRoot`	Will search for the optimal root position using the top matches from root-to-tip regression. If searchRoot=x, dates will be estimated for x trees, and the estimate with the highest likelihood will be returned.
`quiet`	If TRUE, will suppress messages during execution
`temporalConstraints`	If TRUE, will enforce the condition that an ancestor node in the phylogeny occurs before all progeny. Equivalently, this will preclude negative branch lengths. Note that execution is faster if this option is FALSE.
`clock`	The choice of molecular clock model. Choices are 'strict'(default), 'uncorrelated', or 'additive'
`estimateSampleTimes`	If some sample times are not known with certainty, bounds can be provided with this option. This should take the form of a data frame with columns 'lower' and 'upper' providing the sample time bounds for each uncertain tip. Row names of the data frame should correspond to elements in tip.label of the input tree. Tips with sample time bounds in this data frame do not need to appear in the *sts* argument, however if they are included in *sts*, that value will be used as a starting condition for optimisation.

`estimateSampleTimes_densities`	An optional named list of log densities which would be used as priors for unknown sample times. Names should correspond to elements in tip.label with uncertain sample times.
`numStartConditions`	Will attempt optimisation from more than one starting point if >0
`clsSolver`	Which package should be used for constrained least-squares? Options are "mgcv" or "limSolve"
`meanRateLimits`	Optional constraints for the mean substitution rate
`ncpu`	Number of threads for parallel computing
`parallel_foreach`	If TRUE, will use the "foreach" package instead of the "parallel" package. This may work better on some HPC systems.

Details

Estimates the calendar time of nodes in the given phylogenetic tree with branches in units of substitutions per site. The calendar time of each sample must also be specified and the length of the sequences used to estimate the tree. If the tree is not rooted, this function will estimate the root position. For an introduction to all options and features, see the vignette on Influenza H3N2: vignette("h3n2")

Multiple molecular clock models are supported including a strict clock and two variations on relaxed clocks. The 'uncorrelated' relaxed clock is the Gamma-Poisson mixture presented by Volz and Frost (2017), while the 'additive' variance model was developed by Didelot & Volz (2019).

Value

A time-scaled tree and estimated molecular clock rate

References

E.M. Volz and Frost, S.D.W. (2017) Scalable relaxed clock phylogenetic dating. Virus Evolution.
X. Didelot and Volz, E.M. (2019) Additive uncorrelated relaxed clock models.

Author(s)

Erik M Volz <erik.volz@gmail.com>

See Also

ape::chronos ape::estimate.mu

Examples

```
## simulate a random tree and sample times for demonstration
# make a random tree:
tre <- ape::rtree(50)
# sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
```

```
# modify edge length to represent evolutionary distance with rate 1e-3:
tre$edge.length <- tre$edge.length * 1e-3
# treedater:
td <- dater( tre, sts =sts , s = 1000, clock='strict', omega0=.0015)
```

gibbs_jitter — *Sample node dates conditional on root time using Gibbs sampling and date prior*

Description

This function is useful for 'smoothing out' time trees that have many adjacent small branch lengths (essential polytomies). It returns a list of smoothed trees.

Usage

```
gibbs_jitter(
  dtr,
  iter = 1000,
  burn_pc = 20,
  returnTrees = 10,
  res = 100,
  report = 10
)
```

Arguments

dtr	A treedater fit
iter	Number of iterations (every node time is resampled in each iteration)
burn_pc	Remove this proportion as burnin
returnTrees	Integer number of trees to return.
res	Time resolution for proposing new node times
report	Report progress after this many iterations. Set to Inf to turn it off
return	A list of treedater trees

Examples

```
## Not run:
# make a random tree:
tre <- ape::rtree(50)
# sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
# modify edge length to represent evolutionary distance with rate 1e-3:
tre$edge.length <- tre$edge.length * 1e-3
# treedater:
```

```
td <- dater( tre, sts =sts, clock='strict', s = 1000, omega0=.0015 )
gibbs_jitter( td )

## End(Not run)
```

---

goodnessOfFitPlot *Produce a goodness of fit plot*

Description

The sorted tail probabilties (p values) for each edge in the tree under the fitted model

Usage

```
goodnessOfFitPlot(td)
```

Arguments

td	A treedater object generated by the `dater` function

---

outlierLineages *Detect lineages with unusually large evolutionary divergence under the fitted treedater model*

Description

Outliers are detected using the *stats::p.adjust* function and the 'fdr' function. The test requires that *dater* was used with the temporalConstraints=TRUE.

Usage

```
outlierLineages(td, alpha = 0.05, type = c("tips", "internal", "all"))
```

Arguments

td	A fitted treedater object
alpha	The tail probability used for classifying lineages as outliers
type	Should outliers be detected on tip lineages, interal lineages, or all lineages?

Value

A data frame summarizing for each lineage the p values, adjusted p values ('q'), likelihood, rates, and branch lengths.

See Also

dater outlier.tips

outlierTips — *Detect terminal lineages with unusually large evolutionary divergence under the fitted treedater model*

Description

This is a convient wrapper of the *outlier.lineages*

Usage

```
outlierTips(td, alpha = 0.05)
```

Arguments

`td`	A fitted treedater object
`alpha`	The tail probability used for classifying lineages as outliers

Value

A data frame summarizing for each lineage the p values, adjusted p values ('q'), likelihood, rates, and branch lengths.

See Also

dater outlier.lineages

parboot — *Estimate of confidence intervals using parametric bootstrap for molecular clock dating.*

Description

This function simulates phylogenies with branch lengths in units of substitutions per site. Simulations are based on a fitted treedater object which provides parameters of the molecular clock model. The treedater method is applied to each simulated tree providing a Monte Carlo estimate of variance in rates and dates.

Usage

```
parboot(
  td,
  nreps = 100,
  ncpu = 1,
  overrideTempConstraint = TRUE,
  overrideClock = NULL,
  overrideSearchRoot = TRUE,
  overrideSeqLength = NULL,
  quiet = TRUE,
  normalApproxTMRCA = FALSE,
  parallel_foreach = FALSE
)
```

Arguments

`td`	A fitted treedater object
`nreps`	Integer number of simulations to be carried out
`ncpu`	Number of threads to use for parallel computation. Recommended.
`overrideTempConstraint`	If TRUE (default) will not enforce positive branch lengths in simualtion replicates. Will speed up execution.
`overrideClock`	May be 'strict' or 'additive' or 'uncorrelated' in which case will force simulations to fit the corresponding model. If ommitted, will inherit the clock model from td
`overrideSearchRoot`	If TRUE, will re-use root position from input treedater tree. Otherwise may re-estimate root position in simulations
`overrideSeqLength`	Optional sequence length to use in simulations
`quiet`	If TRUE will minimize output printed to screen
`normalApproxTMRCA`	If TRUE will use estimate standard deviation from simulation replicates and report confidence interval based on normal distribution
`parallel_foreach`	If TRUE will use the foreach package for parallelization. May work better on HPC systems.

Details

If the original treedater fit estimated the root position, root position will also be estimated for each simulation, so the returned trees may have different root positions. Some replicates may converge to a strict clock or a relaxed clock, so the parameter estimates in each replicate may not be directly comparable. It is possible to compute confidence intervals for the times of particular nodes or for estimated sample times by inspecting the output from each fitted treedater object, which is contained in the $trees attribute.

Value

A list with elements

- trees: The fitted treedater objects corresponding to each simulation
- meanRates: Vector of estimated rates for each simulation
- meanRate_CI: Confidence interval for substitution rate

- coef_of_variation_CI: Confidence interval for rate variation
- timeOfMRCA_CI: Confidence interval for time of common ancestor

Author(s)

Erik M Volz <erik.volz@gmail.com>

See Also

dater boot

Examples

```
# make a random tree
tre <- ape::rtree(25)
# simulate sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
# modify edge length to represent evolutionary distance with rate 1e-3:
tre$edge.length <- tre$edge.length * 1e-3
# treedater:
td <- dater( tre, sts=sts, s=1000, clock='strict', omega0=.0015 )
# parametric bootstrap:
pb <- parboot( td, nreps=25 )
# plot lineages through time
plot( pb )
```

`plot.bootTreedater` *Plots lineages through time and confidence intervals estimated by bootstrap.*

Description

Plots lineages through time and confidence intervals estimated by bootstrap.

Usage

```
## S3 method for class 'bootTreedater'
plot(x, t0 = NA, res = 100, ggplot = FALSE, cumulative = FALSE, ...)
```

Arguments

`x`	A bootTreedater object produced by *parboot* or *boot*
`t0`	The lower bound of the time axis to show
`res`	The number of time points on the time axis
`ggplot`	If TRUE, will return a plot made with the ggplot2 package
`cumulative`	If TRUE, will show only decreasing lineages through time
`...`	Additional arg’s are passed to *ggplot* or *plot*

relaxedClockTest — *Use parametric bootstrap to test if relaxed clock offers improved fit to data.*

Description

This function simulates phylogenies with branch lengths in units of substitutions per site. Simulations are based on a fitted treedater object which provides parameters of the molecular clock model. The coefficient of variation of rates is estimated using a relaxed clock model applied to strict clock simulations. Estimates of the CV is then compared to the null distribution provided by simulations.

Usage

```
relaxedClockTest(..., nreps = 100, overrideTempConstraint = T, ncpu = 1)
```

Arguments

`...`	arguments passed to *dater*
`nreps`	Integer number of simulations
`overrideTempConstraint`	see *parboot*
`ncpu`	Number of threads to use for parallel computation. Recommended.

Details

This function will print the optimal clock model and the distribution of the coefficient of variation statistic under the null hypothesis (strict clock). Parameters passed to this function should be the same as when calling *dater*.

Value

A list with elements:

- strict_treedater: A dater object under a strict clock
- relaxed_treedater: A dater object under a relaxed clock
- clock: The favoured clock model
- parboot: Result of call to *parboot* using fitted treedater and forcing a relaxed clock
- nullHypothesis_coef_of_variation_CI: The null hypothesis CV

Author(s)

Erik M Volz <erik.volz@gmail.com>

Examples

```
# simulate a tree
tre <- ape::rtree(25)
# sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
# modify edge length to represent evolutionary distance with rate 1e-3:
tre$edge.length <- tre$edge.length * 1e-3
relaxedClockTest( tre, sts, s= 1000,  omega0=.0015 , nreps=25)
```

`rootToTipRegressionPlot`

Plot evolutionary distance from root to sample times and estimated internal node times and regression lines

---

Description

If a range of sample times was given, these will be estimated. Red and black respectively indicate sample and internal nodes. This function will print statistics computed from the linear regression model.

Usage

```
rootToTipRegressionPlot(
  td,
  show.tip.labels = FALSE,
  textopts = NULL,
  pointopts = NULL,
  ...
)
```

Arguments

`td`	A fitted treedater object
`show.tip.labels`	If TRUE, the names of each sample will be plotted at the their corresponding time and evoutionary distance
`textopts`	An optional list of parameters for plotted tip labels. Passed to the *text* function.
`pointopts`	An optional list of parameters for plotted points if showing tip labels. Passed to the *points* function.
`...`	Additional arguments are passed to plot

Value

The fitted linear model (class 'lm')

Examples

```
## simulate a random tree and sample times for demonstration
# make a random tree:
tre <- ape::rtree(50)
# sample times based on distance from root to tip:
sts <- setNames( ape::node.depth.edgelength( tre )[1:ape::Ntip(tre)], tre$tip.label)
# modify edge length to represent evolutionary distance with rate 1e-3:
tre$edge.length <- tre$edge.length * 1e-3
# treedater:
td <- dater( tre, sts =sts, clock='strict', s = 1000, omega0=.0015 )
# root to tip regression:
fit = rootToTipRegressionPlot( td )
summary(fit)
```

---

sampleYearsFromLabels *Compute a vector of numeric sample times from labels in a sequence aligment or phylogeny*

Description

Compute a vector of numeric sample times from labels in a sequence aligment or phylogeny

Usage

```
sampleYearsFromLabels(
  tips,
  dateFormat = "%Y-%m-%d",
  delimiter = NULL,
  index = NULL,
  regex = NULL
)
```

Arguments

`tips`	A character vector supplying the name of each sample
`dateFormat`	The format of the sample date. See ?Date for more information
`delimiter`	Character(s) which separate data in each label
`index`	Integer position of the date string in each label with respect to *delimiter*
`regex`	A regular expression for finding the date substring. Should not be used with *delimiter* or *index*

Value

Numeric vector with sample time in decimal format.

Examples

```
## A couple of labels for Ebola virus sequences:
sampleYearsFromLabels( c('EBOV|AA000000|EM104|SierraLeone_EM|2014-06-02'
                       , 'EBOV|AA000000|G3713|SierraLeone_G|2014-06-09')
, delimiter='|' )
## Equivalently:
sampleYearsFromLabels( c('EBOV|AA000000|EM104|SierraLeone_EM|2014-06-02'
                       , 'EBOV|AA000000|G3713|SierraLeone_G|2014-06-09')
 , regex='[0-9]+-[0-9]+-[0-9]+')
```

Index