Package 'treestructure'

July 5, 2026

Type Package

Title Detect Population Structure Within Phylogenetic Trees

Version 1.0.1

Date 2026-07-03

Description Algorithms for detecting population structure from the history of coalescent events recorded in phylogenetic trees. This method classifies each tip and internal node of a tree into disjoint sets characterized by similar coalescent patterns.

License GPL (>= 2)

Suggests ggtree, ggplot2, knitr, rmarkdown, getopt, phangorn, treeio, testthat (>= 3.0.0)

Depends R (>= 4.1.0)

Imports ape (>= 5.0), rlang

LinkingTo Rcpp

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.1

Encoding UTF-8

URL https://emvolz-phylodynamics.github.io/treestructure/, https://github.com/emvolz-phylodynamics/treestructure

BugReports https://github.com/emvolz-phylodynamics/treestructure/issues

NeedsCompilation yes

Author Erik Volz [aut, cre] (ORCID: <https://orcid.org/0000-0001-6268-8937>), Fabricia F. Nascimento [ctb] (ORCID: <https://orcid.org/0000-0001-9426-6835>), Vinicius B. Franceschi [ctb] (ORCID: <https://orcid.org/0000-0002-0006-9337>)

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

Repository CRAN

Date/Publication 2026-07-05 10:50:03 UTC

Contents

addtips 2
plot.TreeStructure 3
treestructure.test 4
trestruct 5

Index **8**

`addtips` *Compare and add tips into new treestructure object*

Description

Compares a new input tree to an old treestructure fit and merges tips into a new treestructure object. Tips in the new tree that are not in the new treestructure will be merged. Merging is carried out based on a phylogenetic criterion. The new tips are added to the cluster which shares its MRCA (most recent common ancestor).

Usage

```
addtips(trst, tre)
```

Arguments

`trst`	Original treestructure fit that that will be updated.
`tre`	A new tree (ape::phylo) which may contain samples not in trst. This tree must be rooted, but does not need to be time-scaled or binary.

Value

A new treestructure fit.

Author(s)

Erik Volz

Examples

```
set.seed(072023)
# simulate two trees and bind them to simulate structure
tr1 <- ape::rcoal( 50 )
tr2 <- ape::rcoal( 100 )
tr1$tip.label <- gsub(tr1$tip.label, patt = 't', rep = 's')
tr1$edge.length <- tr1$edge.length*.5
tr1$root.edge <- 1
tr2$root.edge <- 1
tr <- ape::bind.tree(tr1, tr2, position = .5 ) |> ape::multi2di()
```

```
# subsample the tree to simulating missing tips and estimate structure
ex <- sample( tr$tip.label, size = 30, replace = FALSE)
tr0 <- ape::drop.tip( tr, ex )
(s0 <- treestructure::trestruct( tr0 ))

# assign structure to the previously missing tips
(s <- treestructure::addtips( s0, tr ))
```

`plot.TreeStructure` *Plot TreeStructure tree with cluster and partition variables*

Description

Plot TreeStructure tree with cluster and partition variables

Usage

```
## S3 method for class 'TreeStructure'
plot(x, use_ggtree = TRUE, ...)
```

Arguments

`x`	A TreeStructure object
`use_ggtree`	Toggle ggtree or ape plotting behavior
`...`	Additional arguments passed to ggtree or ape::plot.phylo

Examples

```
#tree <- ape::read.tree( system.file('sim.nwk', package = 'treestructure') )
# you can run the example below before plotting
#struc <- trestruct( tree )

#because it can take a minute or so to run treestructure, we will load it here
struc <- readRDS( system.file('struc_plot_example.rds', package='treestructure') )
#plot treestructure object

suppressWarnings(plot(struc))
```

`treestructure.test` *Test treestructure hypothesis*

Description

Test the hypothesis that two clades within a tree were generated by the same coalescent process.

Usage

```
treestructure.test(tre, x, y, nsim = 10000, method = "analytic")
```

Arguments

`tre`	An ape::phylo tree, must be binary and rooted
`x`	A character vector of tip labels or numeric node numbers. If numeric, can include internal node numbers.
`y`	as x, but must be disjoint with x
`nsim`	Number of simulations for the null distribution. Only used when `method = 'sim'` (larger = slower and more accurate).
`method`	How the null distribution of the rank-sum statistic is characterised. `'analytic'` (the default) computes the exact mean and variance of the coalescent null with a deterministic recursion and reports a normal-approximation p-value; `'sim'` uses Monte-Carlo simulation.

Examples

```
tree <- ape::read.tree( system.file('sim.nwk', package = 'treestructure') )

# you can run the example below before running test
#struc <- trestruct( tree )

#because it can take a minute or so to run treestructure, we will load it here
struc <- readRDS( system.file('struc_plot_example.rds', package='treestructure') )

#run the test

results <- treestructure.test(tree, x = struc$clusterSets[[1]],
                              y = struc$clusterSets[[2]])

print(results)
```

`trestruct` *Detect cryptic population structure in time trees*

Description

Estimates a partition of a time-scaled tree by contrasting coalescent patterns.

Usage

```
trestruct(
  tre,
  fdr = 0.2,
  level = 0.01,
  minCladeSize = 10,
  nodeSupportValues = FALSE,
  nodeSupportThreshold = 95,
  minOverlap = -Inf,
  nsim = 10000,
  ncpu = 1,
  verbosity = 1,
  debugLevel = 0,
  levellb = 0.001,
  levelub = 0.1,
  res = 11,
  method = "analytic",
  split = c("bonferroni", "bh")
)
```

Arguments

`tre`	A tree of type ape::phylo. Must be rooted. If the tree has multifurcations, it will be converted to a binary tree before processing.
`fdr`	Target false discovery rate for detected structure, a number in (0,1). This is the default way of choosing the split threshold (`fdr = 0.2`): the threshold at each scan is calibrated so that the whole-tree false discovery rate is controlled at this level (see details for the precise meaning). It is analytic and requires no simulation. An explicitly supplied `level` takes precedence unless `fdr` is also given; set `fdr = NULL` to use `level` instead.
`level`	Significance level for finding a new split within a set of tips. Used when `fdr = NULL`, or when `level` is supplied without an explicit `fdr`. This is a subjective clustering threshold, not an error rate; prefer `fdr`. Can also be NULL, in which case the optimal level is found according to the CH index (see details).
`minCladeSize`	All clusters within partition must have at least this many tips.
`nodeSupportValues`	Node support values such as produced by bootstrap or Bayesian credibility scores. Must be logical or vector with length equal to number of internal nodes

in the tree. If nodeSupportValues = TRUE, then the function will get the information on node support from the tree. If numeric vector, these values should be between 0 and 100.

`nodeSupportThreshold`
: Threshold node support value between 0 and 100. Nodes with support lower than this threshold will not be tested.

`minOverlap`
: Threshold time overlap required to find splits in a clade.

`nsim`
: Number of simulations for computing null distribution of test statistics. Only used when `method = 'sim'`.

`ncpu`
: If > 1 will compute statistics in parallel using multiple CPUs.

`verbosity`
: If > 0 will print information about progress of the algorithm.

`debugLevel`
: If > 0 will produce additional data in return value.

`levellb`
: If optimizing the 'level' parameter, this is the lower bound for the search.

`levelub`
: If optimizing the 'level' parameter, this is the upper bound for the search.

`res`
: If optimizing the 'level' parameter, this is the number of values to test.

`method`
: How the coalescent null of the rank-sum statistic is characterised at each test. `'analytic'` (the default) computes the exact mean and variance by a deterministic recursion (fast, and deterministic so results are reproducible); `'sim'` uses Monte-Carlo simulation with `nsim` replicates (the original behaviour).

`split`
: Multiple-testing correction applied at each scan in `fdr` mode. `'bonferroni'` (the default) uses a per-scan Bonferroni bound; `'bh'` uses a Benjamini-Hochberg step-up, which is less conservative and retains more power on large trees with abundant moderate structure (where the Bonferroni bound to make the first split can grow with the tree size). Ignored when a subjective `level` is used.

Details

Estimates a partition of a time-scaled tree by contrasting coalescent patterns. The algorithm is premised on a Kingman coalescent null hypothesis for the ordering of node heights when contrasting two clades, and a test statistic is formulated based on the rank sum of node times in the tree. If node support values are available (as computed by bootstrap procedures), the method can optionally exclude designation of structure on poorly supported nodes. The method will not designate structure on nodes with zero branch length relative to their immediate ancestor. The significance level for detecting significant partitions of the tree can be provided, or a range of values can be examined. The CH index based on within- and between-cluster variance in node heights can be used to select a significance level if none is provided.

Calibrating to a false discovery rate (the default). By default, and whenever `fdr` is supplied, the split threshold is calibrated to a target false discovery rate rather than to a subjective `level`. At each scan the algorithm splits at the most extreme eligible candidate clade only if its standardised statistic clears the Bonferroni threshold $\Phi^{-1}(1 - fdr/(2k))$, where k is the number of *eligible* candidates in that scan; candidates excluded by `minCladeSize`, node support, or time overlap do not count towards k. The `fdr` refers to the **whole tree**, not an individual scan or clade: under the global null of one unstructured coalescent the probability of designating *any* structure equals `fdr`, and when real structure is present `fdr` bounds the expected fraction of spurious splits among all splits. This whole-tree guarantee is obtained by controlling each scan; it is not a per-clade p-value. Unlike `level`, the analytic default requires no simulation and is deterministic.

The returned object also carries a global-null test in `$global.test` (the root-scan $\max|z|$, the number of candidates k, and a Bonferroni p-value for the presence of *any* structure), and, in `fdr` mode, a heterochronous-sampling diagnostic in `$hetero`. Serially sampled (heterochronous) trees can modestly inflate the realised FDR; this is reported and discussed in the package vignette.

Value

A TreeStructure object which includes cluster and partition assignment for each tip of the tree.

References

Volz EM, Carsten W, Grad YH, Frost SDW, Dennis AM, Didelot X. Identification of hidden population structure in time-scaled phylogenies. Systematic Biology 2020; 69(5):884-896.

Author(s)

Erik M Volz

Examples

```
tree <- ape::rcoal(50)
# subjective clustering threshold (default):
struct <-  trestruct( tree )
print(struct)
# calibrate the threshold to a target false discovery rate instead:
struct_fdr <- trestruct( tree, fdr = 0.05 )
print(struct_fdr)
```

Index

addtips, 2

plot.TreeStructure, 3

treestructure.test, 4
trestruct, 5