Package 'ConsTree'

September 15, 2026

Title Fast Algorithms for Phylogenetic Consensus Trees

Version 1.0.0

License GPL (>= 3)

Copyright Incorporates C++ code from 'FACT' (GPL-3, doi:10.1145/2925985), 'FACT2' and 'FDCT', by kind permission of Jesper Jansson and colleagues.

Description Efficient construction of phylogenetic consensus trees. Methods include strict, majority-rule, majority-rule (+), loose (combinable component / semi-strict), greedy, Adams, frequency difference, R*, and local consensus.

URL https://constree.github.io/

BugReports https://github.com/ms609/ConsTree/issues/

Depends R (>= 4.1.0),

Imports ape (>= 5.6), Rcpp (>= 1.0.0), Rdpack (>= 2.6), TreeTools (>= 2.3.0),

LinkingTo Rcpp, TreeTools (>= 2.1.0),

SystemRequirements C++17

Suggests knitr, Quartet, rmarkdown, spelling, testthat (>= 3.0), TreeSearch (>= 2.0.0),

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel false

Config/roxygen2/version 8.0.0

Config/Needs/memcheck pkgdown, testthat

RdMacros Rdpack

Encoding UTF-8

Language en-GB

NeedsCompilation yes

Author Martin R. Smith [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-5660-1727>),
Jesper Jansson [cph] (Consensus algorithms (FACT, FACT2, FDCT)),
Chuanqi Shen [cph] (Consensus algorithms (FACT)),
Wing-Kin Sung [cph] (Consensus algorithms (FACT, FACT2, FDCT))

Maintainer Martin R. Smith <martin.smith@durham.ac.uk>

Repository CRAN

Date/Publication 2026-09-15 12:10:19 UTC

Contents

`Adams` *Adams consensus tree*

Description

`Adams()` computes the Adams (1972) consensus tree.

Usage

```
Adams(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Details

The Adams consensus places each species in the the smallest group to which it belongs on all input trees. Consequently, it may contain groupings that do not appear in any input tree#' This implementation builds on the algorithm of (Jansson et al. 2017); please cite this paper where you use this method.

Value

`Adams()` returns an object of class `phylo` denoting the Adams consensus tree.

References

Adams EN (1972). "Consensus techniques and the comparison of taxonomic trees." *Systematic Zoology*, **21**(4), 390–397. doi:10.2307/2412432.

Jansson J, Li Z, Sung W (2017). "On finding the Adams consensus tree." *Information and Computation*, **256**, 334–347. doi:10.1016/j.ic.2017.08.002.

See Also

Other consensus methods: `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
# Two rooted trees that disagree only on the position of one leaf
trees <- c(ape::read.tree(text = "(((a, b), c), d);"),
           ape::read.tree(text = "(((a, b), d), c);"))
# keeps the clade (a, b); leaves c, d unresolved at the root
ape::write.tree(Adams(trees))
```

Average *Average consensus tree*

Description

`Average()` returns the average consensus (Lapointe and Cucumel 1997): the tree whose path-length (patristic) distances most closely match the average of the path-length distances of the input trees.

Usage

```
Average(
  trees,
  method = c("fastme.bal", "ls", "nj", "bionj", "fastme.ols"),
  weights = NULL,
  scale = c("none", "max"),
  edgeLengths = NA,
```

```
  outgroup = NULL,
  check.labels = TRUE,
  lsControl = list()
)
```

Arguments

trees	A list of trees, or a multiPhylo object. All entries must share the same leaf labels.
method	Character specifying how to build the tree from the average distance matrix: • "fastme.bal" (the default) returns the balanced minimum-evolution tree (Desper and Gascuel 2002): a fast, accurate approximation of the least-squares tree; • "ls" searches for the least-squares tree using TreeSearch::LeastSquaresTree(), a non-negative least-squares NNI/SPR search; • "nj", "bionj" and "fastme.ols" return the corresponding distance tree (Saitou and Nei 1987; Gascuel 1997).
weights	Numeric vector specifying the weight of each tree in the average (e.g. posterior probabilities), with one entry per tree. If NULL, each tree is weighted equally.
scale	Character specifying whether to rescale each tree's distance matrix before averaging. "none" leaves matrices unscaled; "max" divides each matrix by its largest entry, recommended when absolute distances are not comparable (Lapointe and Cucumel 1997).
edgeLengths	Logical specifying whether to use edge lengths when computing path-length distances. TRUE requires edge lengths; FALSE does not use edge lengths; NA uses edge lengths when all trees have them, and otherwise counts edges.
outgroup	Character vector specifying tip label(s) on which to root the result. NULL returns an unrooted tree.
check.labels	Logical specifying whether to confirm that every tree contains the same leaves.
lsControl	Named list of further arguments for the least-squares search (method = "ls"), passed to TreeSearch::LeastSquaresTree().

Details

Because the average of several path-length matrices is usually not itself realisable by any tree, Average() approximates it: by default, with the fast balanced minimum-evolution tree; or if method = "ls", by NP-hard least-squares search (Day 1987).

Value

Average() returns an object of class phylo with fitted branch lengths denoting the average consensus tree.

References

Day WHE (1987). "Computational complexity of inferring phylogenies from dissimilarity matrices." *Bulletin of Mathematical Biology*, **49**(4), 461–467. doi:10.1007/BF02458863.

Desper R, Gascuel O (2002). "Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle." *Journal of Computational Biology*, **9**(5), 687–705. doi:10.1089/106652702761034136.

Gascuel O (1997). "BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data." *Molecular Biology and Evolution*, **14**(7), 685–695. doi:10.1093/oxfordjournals.molbev.a025808.

Lapointe F, Cucumel G (1997). "The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa." *Systematic Biology*, **46**(2), 306–312. doi:10.1093/sysbio/46.2.306.

Saitou N, Nei M (1987). "The neighbor-joining method: a new method for reconstructing phylogenetic trees." *Molecular Biology and Evolution*, **4**(4), 406–425. doi:10.1093/oxfordjournals.molbev.a040454.

See Also

Split-based summaries: `Strict()`, `Majority()`.

Other consensus methods: `Adams()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::rmtree(5, 8)    # five random eight-leaf trees
Average(trees)                # fast (balanced minimum evolution) default

if (requireNamespace("TreeSearch", quietly = TRUE) &&
    exists("LeastSquaresTree", where = asNamespace("TreeSearch"),
           mode = "function")) {
  Average(trees, method = "ls")    # faithful least-squares fit (slower)

  # use Fitch-Margoliash weighting:
  Average(trees, method = "ls",
    lsControl = list(spr = FALSE, maxHits = 5L, weight = "fm")
  )
}
```

`BHVDistance` *Geodesic (BHV) distance between trees*

Description

`BHVDistance()` returns the geodesic distance between phylogenetic trees with edge lengths in the Billera-Holmes-Vogtmann (BHV) treespace (Billera et al. 2001), computed with the polynomial-time GTP algorithm of Owen and Provan (2011).

Usage

```
BHVDistance(tree1, tree2 = NULL)

BHVDist(tree1, tree2 = NULL)

BHV(tree1, tree2 = NULL)
```

Arguments

`tree1`, `tree2`	A `phylo` tree, a `multiPhylo` object, or a list of `phylo` trees, each carrying `edge.length`. All trees across both arguments must share the same leaf labels. `tree2` may be omitted when `tree1` is a collection, in which case all pairwise distances within `tree1` are returned.

Value

`BHVDistance()` returns distances between the input trees:

- If both inputs are single trees, the return value is a single non-negative number.
- When comparing a tree to a collection of trees, the return value is a named numeric vector with one entry per tree in the collection.
- If `tree2` is omitted, or is identical to `tree1`, the return value is a `stats::dist` object of all pairwise distances.
- If `tree1` and `tree2` are different collections, the return value comprises a numeric matrix with rows corresponding to `tree1` and columns to `tree2`.

References

Billera LJ, Holmes SP, Vogtmann K (2001). "Geometry of the space of phylogenetic trees." *Advances in Applied Mathematics*, **27**(4), 733–767. doi:10.1006/aama.2001.0759.

Owen M, Provan JS (2011). "A fast algorithm for computing geodesic distances in tree space." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **8**(1), 2–13. doi:10.1109/TCBB.2010.3.

See Also

Other BHV summaries: `BHVMean()`

Examples

```
set.seed(2)
trees <- lapply(1:4, function(i)
  TreeTools::RandomTree(8, root = TRUE, lengths = runif)
)
t1 <- trees[[1]]
t2 <- trees[[2]]

BHVDistance(t1, t2)                 # scalar
BHVDistance(t1, trees)              # named vector
BHVDistance(trees)                  # dist (pairwise)
BHVDistance(trees[1:2], trees[3:4])  # matrix
```

BHVMean *Fréchet mean and variance in BHV treespace*

Description

`BHVMean()` returns the Fréchet (Karcher) mean of a set of trees in BHV treespace: the tree that minimizes the sum of squared geodesic distances to the sample (Brown and Owen 2020). The mean is approximated by the iterative law-of-large-numbers algorithm of Sturm (2003) and Miller et al. (2015): starting from a sample tree, each step walks $\frac{1}{k+1}$ of the way along the geodesic towards a randomly chosen sample tree.

Usage

```
BHVMean(trees, tolerance = 1e-04, maxIter = 100000L, cauchyLength = 10L)

BHVVariance(trees, mean = NULL, type = c("average", "sum"))
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object; all entries must share the same leaf labels and carry `edge.length`.
`tolerance`	Numeric stating convergence threshold relative to the sample standard deviation. Iteration stops once `cauchyLength` consecutive steps each move the estimate less than `tolerance` times the sample standard deviation.
`maxIter`	Integer specifying the maximum number of iterations.
`cauchyLength`	Integer specifying the number of consecutive small steps required to declare convergence.
`mean`	Object of class `phylo` specifying a pre-computed mean tree; computed via `BHVMean()` if `NULL`.
`type`	Character specifying whether to return the mean squared distance (`"average"`) or the total squared distance (`"sum"`).

Details

`BHVVariance()` returns the Fréchet variance: by default the mean squared geodesic distance from the sample to its mean, $\frac{1}{r}\sum_i d(\bar{T}, T_i)^2$; with `type = "sum"`, the total $\sum_i d(\bar{T}, T_i)^2$.

The mean is "sticky": perturbing one sample tree need not move it, and it is pulled towards lower-dimensional (less resolved) orthants, so it may be unresolved even when the sample trees are binary (Brown and Owen 2020).

Value

`BHVMean()` returns an object of class `phylo` denoting the mean tree, with attributes `iterations` (integer; number of steps taken) and `converged` (logical). Because the step length shrinks as $1/(k+1)$, `converged = TRUE` indicates that successive estimates have stopped moving appreciably (the stopping rule was met before `maxIter`), not a guaranteed bound on the distance to the exact mean; tighten `tolerance` for greater precision.

`BHVVariance()` returns a single non-negative number.

References

Brown DG, Owen M (2020). "Mean and variance of phylogenetic trees." *Systematic Biology*, **69**(1), 139–154. doi:10.1093/sysbio/syz041.

Miller E, Owen M, Provan JS (2015). "Polyhedral computational geometry for averaging metric phylogenetic trees." *Advances in Applied Mathematics*, **68**, 51–91. doi:10.1016/j.aam.2015.04.002.

Sturm K (2003). "Probability measures on metric spaces of nonpositive curvature." In *Heat Kernels and Analysis on Manifolds, Graphs, and Metric Spaces*, volume 338 of *Contemporary Mathematics*, 357–390. doi:10.1090/conm/338/06080.

See Also

Other BHV summaries: `BHVDistance()`

Examples

```
set.seed(0)
trees <- lapply(1:25, function(i) {
  tree <- TreeTools::RandomTree(6, root = FALSE)
  tree$edge.length <- runif(nrow(tree$edge))
  tree
})
meanTree <- BHVMean(trees)
BHVVariance(trees, mean = meanTree)
```

`Frequency` *Frequency-difference consensus tree*

Description

`Frequency()` computes the frequency-difference consensus, which retains each split that occurs more often than every split that conflicts with it.

Usage

```
Frequency(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Details

The frequency-difference consensus is at least as resolved as the majority-rule consensus (`Majority()`), and is contained within the greedy consensus (`Greedy()`).

This implementation builds on the FDCT algorithm of Jansson et al. (2024); please cite that paper when using this method.

Value

`Frequency()` returns an object of class `phylo` denoting the frequency consensus, rooted as in the first entry of `trees`.

References

Jansson J, Sung W, Tabatabaee SA, Yang Y (2024). "A Faster Algorithm for Constructing the Frequency Difference Consensus Tree." In Beyersdorff O, Kanté MM, Kupferman O, Lokshtanov D (eds.), *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 43:1–43:17. doi:10.4230/LIPIcs.STACS.2024.43.

See Also

Closely related: `Majority()`, `MajorityPlus()`, `Greedy()`.

Other consensus methods: `Adams()`, `Average()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::as.phylo(0:5, 8)
Frequency(trees)
```

`Greedy` *Greedy (extended majority-rule) consensus tree*

Description

`Greedy()` computes the greedy consensus, also termed the extended majority-rule consensus (Bryant 2003). Distinct splits are considered in decreasing order of their frequency across the input trees, breaking ties arbitrarily; each is added to the growing consensus if it is compatible with every split already accepted.

Usage

```
Greedy(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Details

The implementation builds upon the `greedyConsensusFast` algorithm of (Jansson et al. 2016); please cite that paper when using this method.

Value

`Greedy()` returns an object of class `phylo` denoting the consensus, rooted as in the first entry of `trees`.

References

Bryant D (2003). "A classification of consensus methods for phylogenetics." In Janowitz MF, Lapointe F, McMorris FR, Mirkin B, Roberts FS (eds.), *Bioconsensus*, volume 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 163–184. American Mathematical Society. doi:10.1090/dimacs/061/11.

Jansson J, Shen C, Sung W (2016). "Improved algorithms for constructing consensus trees." *Journal of the ACM*, **63**(3), 1–24. doi:10.1145/2925985.

See Also

Closely related: `Strict()`, `Majority()`, `Loose()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::as.phylo(0:5, 8)
Greedy(trees)
```

`Local` *Local consensus tree*

Description

`Local()` returns the local consensus (Jansson et al. 2018) of a set of rooted trees. The local consensus is the most conservative tree consistent with the rooted triplets shared by every input tree. The minimum rooted local consensus (MinRLC, `type = "rooted"`) and the minimum induced local consensus (MinILC, `type = "induced"`) differ in how the resolution of the result is scored.

Usage

```
Local(trees, type = c("rooted", "induced"))
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.
`type`	Character specifying whether to compute the minimum rooted local consensus (`"rooted"`, the default; MinRLC) or the minimum induced local consensus (`"induced"`; MinILC).

Details

The implementation builds on the algorithms of (Jansson et al. 2018); please cite that paper when using this method.

Because the algorithm is exponential, `Local()` is limited to `n <= 20` leaves. Running time is faster when input trees are more congruent.

Value

`Local()` returns an object of class `phylo` denoting the local consensus tree. When there is no valid consensus (as no common triplets separate any pair of trees), a star tree is returned.

References

Jansson J, Rajaby R, Sung W (2018). "Minimal phylogenetic supertrees and local consensus trees." *AIMS Medical Science*, **5**(2), 181–203. doi:10.3934/medsci.2018.2.181.

See Also

Closely related: `Strict()`, `Majority()`, `Adams()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
# Two trees that agree on one cherry but disagree on overall topology
t1 <- ape::read.tree(text = "(1,((2,3),4));")
t2 <- ape::read.tree(text = "(1,((2,4),3));")
Local(list(t1, t2), "rooted")  # keeps clade {2,3,4} only
```

Loose *Loose consensus tree*

Description

`Loose()` computes the loose consensus, also known as the semi-strict or combinable-component consensus (Bremer 1990). It contains every split that is not contradicted by any input tree.

Usage

```
Loose(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Details

The loose consensus refines the strict consensus (`Strict()`). In contrast to the majority-rule consensus (`Majority()`), a split present in most trees may still be contradicted by a minority, and so be excluded from the loose consensus; yet a split that occurs in just one tree will be is retained if no other tree contradicts it.

This implementation builds on the `looseConsensusFast` algorithm of (Jansson et al. 2016); please cite that paper when using this method.

Value

`Loose()` returns the consensus tree, an object of class `phylo`, rooted as in the first entry of `trees`.

References

Bremer K (1990). "Combinable component consensus." *Cladistics*, **6**(4), 369–372. doi:10.1111/j.10960031.1990.tb00551.x.

Jansson J, Shen C, Sung W (2016). "Improved algorithms for constructing consensus trees." *Journal of the ACM*, **63**(3), 1–24. doi:10.1145/2925985.

See Also

Closely related: `Strict()`, `Majority()`, `Greedy()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::as.phylo(0:5, 8)
Loose(trees)
```

`Majority` *Majority-rule consensus tree*

Description

`Majority()` is a simple alias of `TreeTools::Consensus()`, which computes the majority-rule consensus (Margush and McMorris 1981): the tree that contains each split that occurs in more than `p` of the input trees.

Usage

```
Majority(trees, p = 0.5)

MajorityRule(trees, p = 0.5)

MR(trees, p = 0.5)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.
`p`	Numeric between 0.5 and 1: the minimum proportion of trees that must contain a split for it to be retained.

Value

`Majority()` returns an object of class `phylo` denoting the majority rule consensus, rooted as in the first entry of `trees`.

References

Margush T, McMorris FR (1981). "Consensus n-trees." *Bulletin of Mathematical Biology*, **43**(2), 239–244. doi:10.1007/BF02459446.

See Also

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::as.phylo(0:5, 8)
Majority(trees, p = 0.6)
```

`MajorityPlus` *Majority-rule (+) consensus tree*

Description

`MajorityPlus()` computes the majority-rule (+) consensus (Jansson et al. 2016), which displays each clade that occurs in more input trees than contradict it.

Usage

```
MajorityPlus(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Value

`MajorityPlus()` returns an object of class `phylo` denoting the majority-plus consensus tree, rooted as in the first entry of `trees`.

References

Jansson J, Shen C, Sung W (2016). "Improved algorithms for constructing consensus trees." *Journal of the ACM*, **63**(3), 1–24. doi:10.1145/2925985.

See Also

Closely related: `Majority()`, `Greedy()`, `Loose()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `Quartet()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
trees <- ape::as.phylo(0:5, 8)
MajorityPlus(trees)
```

`Quartet` *Consensus tree that minimizes quartet distance*

Description

`Quartet()` constructs a consensus tree that minimizes the sum of symmetric quartet distances to a set of input trees, using a greedy add-and-prune heuristic.

Usage

```
Quartet(
  trees,
  init = c("majority", "star", "extended"),
  greedy = c("best", "first")
)
```

Arguments

`trees`	Object of class `multiPhylo` specifying the input trees. All trees must share the same tip labels.
`init`	Character string specifying the initial tree: • `"majority"`: the majority-rule consensus. • `"star"`: a fully unresolved star tree. • `"extended"`: the extended (greedy) majority-rule consensus.
`greedy`	Character string specifying the greedy strategy: • `"best"`: evaluate all candidates and pick the best action at each step. • `"first"`: pick the first improving action encountered (faster; may give a slightly worse result).

Details

Where the majority-rule consensus minimizes the sum of Robinson-Foulds distances to the input trees, `Quartet()` finds an approximate median tree under the symmetric quartet distance (Takazawa et al. 2026), which counts both false-positive and false-negative resolved quartets equally.

Because the quartet distance gives greater weight to deep branches (which resolve more quartets), quartet consensus trees tend to be more resolved than majority-rule trees, especially when phylogenetic signal is low.

The function supports trees with up to 100 tips.

Value

`Quartet()` returns a consensus tree, an object of class `phylo`, unrooted.

References

Takazawa Y, Takeda A, Hayamizu M, Gascuel O (2026). "Outperforming the majority-rule consensus tree using fine-grained dissimilarity measures." *bioRxiv*. doi:10.64898/2026.03.16.712085.

See Also

Closely related: `Strict()`, `Majority()`, `Greedy()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `RStar()`, `Strict()`, `Transfer()`

Examples

```
library("TreeTools", quietly = TRUE)

# Generate bootstrap-like trees
trees <- as.phylo(1:30, nTip = 8)

# Quartet consensus
qc <- Quartet(trees)
plot(qc)

# Compare resolution with majority-rule
mr <- UnrootTree(Consensus(trees, p = 0.5))
cat("Majority-rule splits:", NSplits(mr), "\n")
cat("Quartet consensus splits:", NSplits(qc), "\n")
```

`RStar` *R* consensus tree*

Description

`RStar()` returns the R* consensus (Degnan et al. 2009) of a set of rooted trees.

Usage

```
RStar(trees)
```

Arguments

`trees`	A list of trees, or a `multiPhylo` object. All entries must share the same leaf labels.

Details

The R* consensus is a rooted-triplet method. For every set of three leaves it tallies, across the input trees, the three possible resolved rooted triplets (`ab|c`, `ac|b`, `bc|a`) and keeps whichever appears most frequently. Ties are not kept. The kept triplets form the set of majority resolved triplets, R_{maj}. Then R* is the most resolved tree that displays no resolved triplet outside R_{maj}.

R* is always a refinement of the majority-rule consensus: every majority clade also appears in `RStar()`.

Value

`RStar()` returns the consensus tree, an object of class `phylo`. It is rooted by construction, but when the resolved triplets leave the deepest divergence unresolved the root is a polytomy.

References

Degnan JH, DeGiorgio M, Bryant D, Rosenberg NA (2009). "Properties of consensus methods for inferring species trees from gene trees." *Systematic Biology*, **58**(1), 35–54. doi:10.1093/sysbio/syp008.

See Also

Closely related: `Strict()`, `Majority()`, `Adams()`, `Local()`.

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `Strict()`, `Transfer()`

Examples

```
# Five trees whose majority signal recovers the species tree (((a,b),c),d):
trees <- c(
  ape::read.tree(text = "(((a, b), c), d);"),
  ape::read.tree(text = "(((a, b), c), d);"),
  ape::read.tree(text = "(((a, b), c), d);"),
  ape::read.tree(text = "(((a, c), b), d);"),
  ape::read.tree(text = "(((b, c), a), d);")
)

# (a, b) wins {a,b,c} by plurality (3 vs 1 vs 1)
ape::write.tree(RStar(trees))
```

`Strict` *Strict consensus tree*

Description

`Strict()` returns the strict consensus of a set of trees: the tree that contains exactly those splits (clades) present in every input tree (Day 1985).

Usage

```
Strict(trees)
```

Arguments

trees	A list of trees, or a multiPhylo object. All entries must share the same leaf labels.

Details

This is an alias of TreeTools::Consensus() with p = 1.

Value

Strict() returns the consensus tree, an object of class phylo, rooted as in the first entry of trees.

References

Day WHE (1985). "Optimal algorithms for comparing trees with labeled leaves." *Journal of Classification*, **2**(1), 7–28. doi:10.1007/BF01908061.

See Also

Less conservative summaries: Majority().

Other consensus methods: Adams(), Average(), Frequency(), Greedy(), Local(), Loose(), Majority(), MajorityPlus(), Quartet(), RStar(), Transfer()

Examples

```
trees <- ape::as.phylo(0:5, 8)
Strict(trees)
```

Transfer *Consensus tree minimizing transfer distance*

Description

Construct a consensus tree that minimizes the sum of transfer distances to a set of input trees, using a greedy add-and-prune heuristic.

Usage

```
Transfer(
  trees,
  scale = TRUE,
  greedy = c("best", "first"),
  init = c("empty", "majority")
)
```

Arguments

`trees`	An object of class `multiPhylo`: the input trees. All trees must share the same tip labels.
`scale`	Logical; if `TRUE` (default), use the scaled transfer distance (normalized by light-side size minus one). If `FALSE`, use the unscaled (raw Hamming) transfer distance.
`greedy`	Character string specifying the greedy strategy: `"best"` (default) picks the single highest-benefit action at each step; `"first"` picks the first improving action encountered (faster but potentially lower quality).
`init`	Character string specifying the initial consensus: `"empty"` (default) starts with no splits (purely additive); `"majority"` starts with the majority-rule consensus and refines.

Details

Unlike the majority-rule consensus, which minimizes Robinson-Foulds distance and can be highly unresolved when phylogenetic signal is low, `Transfer()` uses the finer-grained transfer distance (Lemoine et al. 2018) to construct a more resolved consensus tree.

The algorithm pools all splits observed across input trees, computes pairwise transfer distances between them, and greedily adds or removes splits to minimize total transfer dissimilarity cost. The approach follows Takazawa et al. (2026), reimplemented for 'ConsTree' infrastructure.

Value

A tree of class `phylo`.

References

Lemoine F, Domelevo Entfellner J, Wilkinson E, Correia D, Dávila Felipe M, De Oliveira T, Gascuel O (2018). "Renewing Felsenstein's phylogenetic bootstrap in the era of big data." *Nature*, **556**(7702), 452–456. doi:10.1038/s4158601800430.

Takazawa Y, Takeda A, Hayamizu M, Gascuel O (2026). "Outperforming the majority-rule consensus tree using fine-grained dissimilarity measures." *bioRxiv*. doi:10.64898/2026.03.16.712085.

See Also

`Quartet()`

Other consensus methods: `Adams()`, `Average()`, `Frequency()`, `Greedy()`, `Local()`, `Loose()`, `Majority()`, `MajorityPlus()`, `Quartet()`, `RStar()`, `Strict()`

Examples

```
library(TreeTools)
trees <- as.phylo(50:69, nTip = 12)

tc <- Transfer(trees)
mr <- Majority(trees)
```

```
plot(tc, main = "Transfer consensus")
plot(mr, main = "Majority rule", direction = "left")
# Compare resolution
cat("Majority-rule splits:", NSplits(mr), "\n")
cat("Transfer consensus splits:", NSplits(tc), "\n")
```

Index