Package: Quartet (via r-universe)

October 22, 2026

Version 1.4.0

Title Comparison of Phylogenetic Trees Using Quartet and Split Measures

Description Calculates the number of four-taxon subtrees consistent with a pair of cladograms, calculating the symmetric quartet distance of Bandelt & Dress (1986), Reconstructing the shape of a tree from observed dissimilarity data, Advances in Applied Mathematics <doi:10.1016/0196-8858(86)90038-2>, using the tqDist algorithm of Sand et al. (2014), tqDist: a library for computing the quartet and triplet distances between binary or general trees, Bioinformatics <doi:10.1093/bioinformatics/btu157> for pairs of binary trees; calculating the rooted triplet distance using the CPDT algorithm of Jansson & Rajaby (2017), A more practical algorithm for the rooted triplet distance, Journal of Computational Biology <doi:10.1089/cmb.2016.0185>.

URL https://ms609.github.io/Quartet/, https://github.com/ms609/Quartet/

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

Copyright Incorporates code modified from tqDist <doi:10.1093/bioinformatics/btu157> and from CPDT-dist <doi:10.1089/cmb.2016.0185>.

License GPL (>= 2)

Encoding UTF-8

Language en-GB

Depends R (>= 3.6.0), TreeTools (>= 1.4.0),

Imports ape, PlotTools, Rdpack (>= 0.7), Ternary (>= 1.0),

Suggests bookdown, knitr, phangorn, rmarkdown, testthat, TreeDist, usethis, vdiffr,

Config/Needs/check rcmdcheck, tinytex,

Config/Needs/coverage covr

Config/Needs/memcheck pkgdown, testthat

Config/Needs/metadata codemetar

Config/Needs/revdeps revdepcheck

Config/Needs/website pkgdown

Config/roxygen2/version 8.1.0

RdMacros Rdpack

LinkingTo Rcpp, TreeTools,

LazyData true

ByteCompile true

VignetteBuilder knitr

NeedsCompilation yes

Author Martin R. Smith [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-5660-1727>), Jesper Jansson [ctb, cph] (ORCID: <https://orcid.org/0000-0001-6859-8932>), Ramesh Rajaby [ctb, cph], Andreas Sand [ctb, cph], Gerth Stølting Brodal [ctb, cph], Rolf Fagerberg [ctb, cph], Thomas Mailund [ctb, cph], Christian N. S. Pedersen [ctb, cph] (ORCID: <https://orcid.org/0000-0002-8947-6771>), Jens Johansen [ctb, cph], Morten K. Holt [ctb, cph]

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

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

Date/Publication 2026-09-22 14:40:09 UTC

RemoteUrl https://github.com/cran/Quartet

RemoteRef HEAD

RemoteSha f95b6a04f2e8d5241c2b991bf32570fcddb9c3a9

Contents

AllQuartets . . . 3
CompareQuartets . . . 4
CompareQuartetsMulti . . . 5
CompareSplits . . . 6
PairwiseQuartets . . . 8
PlotQuartet . . . 9
QuartetPoints . . . 10
QuartetState . . . 11
ResolvedQuartets . . . 13
SharedQuartetStatus . . . 14
SimilarityMetrics . . . 17
SplitStatus . . . 21
sq_trees . . . 23
SymmetricDifferenceLineEnds . . . 24

TQDist 25
TripletDistance.phylo 26
VisualizeQuartets 29

Index **31**

`AllQuartets` *List all quartets*

Description

Lists all choices of four taxa from a tree.

A more computationally efficient alternative to `combn`.

Usage

```
AllQuartets(nTips)

## S3 method for class 'numeric'
AllQuartets(nTips)

## S3 method for class 'phylo'
AllQuartets(nTips)
```

Arguments

`nTips`	Integer, specifying the number of tips in a tree; or a tree, whose tips will be counted.

Value

`AllQuartets()` returns a matrix with four rows and `choose(n_tips, 4)` columns, with each column corresponding to a unique selection of four different integers less than or equal to `nTips`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

States of quartets in given trees: `QuartetStates()`

Other enumeration: `CompareQuartets()`, `CompareQuartetsMulti()`, `ResolvedQuartets()`

Examples

```
AllQuartets(5)

combn(5, 4) # Provides the same information, but for large
            # values of n_tips is significantly slower.
```

`CompareQuartets` *Compare quartet states by explicit enumeration*

Description

`CompareQuartets()` uses explicit enumeration to compare two lists of quartet states (Estabrook et al. 1985), detailing how many are identical and how many are unresolved. For most purposes, the faster function `QuartetStatus()` will be preferable.

Usage

```
CompareQuartets(x, cf)
```

Arguments

`x`, `cf`	List of quartet states, perhaps generated by `QuartetStates()`.

Value

Returns an array of seven numeric elements, corresponding

N The total number of quartet *statements* for two trees of n leaves, i.e. 2 Q.

Q The total number of quartets for n leaves.

s The number of quartets that are resolved identically in both trees.

d The number of quartets that are resolved differently in each tree.

r1 The number of quartets that are resolved in tree 1, but not in tree 2.

r2 The number of quartets that are resolved in tree 2, but not in tree 1.

u The number of quartets that are unresolved in both trees.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.

See Also

- `QuartetStatus()` generates the same output from a list of trees.

Other element-by-element comparisons: `CompareQuartetsMulti()`, `CompareSplits()`, `PairSharedQuartetStatus()`, `PairwiseQuartets()`, `QuartetState()`, `SharedQuartetStatus()`, `SplitStatus()`

Other enumeration: `AllQuartets()`, `CompareQuartetsMulti()`, `ResolvedQuartets()`

Examples

```
trees <- list(TreeTools::BalancedTree(6),
              TreeTools::PectinateTree(6))
quartets <- QuartetStates(trees)
CompareQuartets(quartets[[1]], quartets[[2]])
```

`CompareQuartetsMulti` *Compare one tree's quartets against others'*

Description

`CompareQuartetsMulti()` counts how many quartets in one tree are resolved in the same way or different ways in a forest of comparison trees.

Usage

```
CompareQuartetsMulti(x, cf)
```

Arguments

`x`	Object of class `phylo` representing the tree of interest.
`cf`	Comparison tree of class `phylo`, or list thereof, each with the same leaves as `x`.

Details

`CompareQuartetsMulti()` explicitly evaluates each quartet in each tree. As such its runtime will increase hyper-exponentially with the number of leaves in trees being compared. 30 leaves will take around 5 seconds; 40 closer to 20 s, and 50 around a minute.

Value

`CompareQuartetsMulti()` returns a named integer vector specifying the number of quartets whose resolution in `x` matches all or any of the resolutions in `cf`.

Named elements are:

N The total number of quartet *statements* for the given number of *n*-leaf trees, i.e. $n_trees \times Q$.

Q The total number of quartets for *n* leaves.

s_all The number of quartets that are resolved identically in all trees.

s_any The number of quartets that are resolved in `x`, and identically in at least one of `cf`.

d_all The number of quartets that are resolved in every tree in `cf`, but never in the same way as they are resolved in in `x`.

d_any The number of quartets in `x` that are resolved differently (i.e. contradicted) in at least one tree in `cf`.

r1_all The number of quartets that are resolved in `x`, but not in any of `cf`.

r1_any The number of quartets that are resolved in `x`, but unresolved in at least one of `cf`.

r2_all The number of quartets that are resolved in all of `cf`, but not in `x`.

r2_any The number of quartets that are resolved in at least one of `cf`, but not in `x`.

u_all The number of quartets that are unresolved in all trees.

u_any The number of quartets that are unresolved in `x` and at least one tree in `cf`.

x_only The number of quartets in `x` that are not resolved the same way in any of `cf`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other element-by-element comparisons: `CompareQuartets()`, `CompareSplits()`, `PairSharedQuartetStatus()`, `PairwiseQuartets()`, `QuartetState()`, `SharedQuartetStatus()`, `SplitStatus()`

Other enumeration: `AllQuartets()`, `CompareQuartets()`, `ResolvedQuartets()`

Examples

```
library("TreeTools")
CompareQuartetsMulti(x  = CollapseNode(as.phylo(42, 6), 8:9),
                     cf = list(BalancedTree(6), PectinateTree(6),
                               CollapseNode(as.phylo(1337, 6), 9:10)))
```

`CompareSplits` *Compare status of splits*

Description

Reports whether splits are present or contradicted in a set of reference splits.

Usage

```
CompareSplits(splits, splits2)

CompareBipartitions(splits, splits2)
```

Arguments

`splits`	An object that can be coerced into class `Splits` using `as.Splits`.
`splits2`	Splits against which to compare `splits`.

Value

A named vector of eight integers, listing the number of unique splits that:

- **N** exist in total; i.e. the number of splits in `splits1` plus the number in `splits2`, equivalent to $2\,s + d1 + d2 + r1 + r2$;
- **P1** occur in `splits1`
- **P2** occur in `splits2`
- **s** occur in both `splits1` and `splits2`;
- **d1** occur in `splits1` but are contradicted by `splits2`;
- **d2** occur in `splits2` but are contradicted by `splits1`;
- **r1** occur in `splits1` only, being neither present in nor contradicted by `splits2`;
- **r2** occur in `splits2` only, being neither present in nor contradicted by `splits1`;
- **RF** occur in one tree only; i.e. $d1 + d2 + r1 + r2$, the Robinson-Foulds distance.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

- Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.
- Robinson DF, Foulds LR (1981). "Comparison of phylogenetic trees." *Mathematical Biosciences*, **53**(1-2), 131–147. doi:10.1016/00255564(81)900432.

See Also

Equivalent function for quartets: `CompareQuartets()`

Other element-by-element comparisons: `CompareQuartets()`, `CompareQuartetsMulti()`, `PairSharedQuartetStatus()`, `PairwiseQuartets()`, `QuartetState()`, `SharedQuartetStatus()`, `SplitStatus()`

Examples

```
splits1 <- TreeTools::BalancedTree(8)
splits2 <- TreeTools::PectinateTree(8)

CompareSplits(splits1, splits2)
```

`PairwiseQuartets` *Pairwise quartet distances*

Description

Computes the quartet distance between each pair of trees in a list.

Usage

```
PairwiseQuartets(trees, Measure = QuartetDivergence)
```

Arguments

`trees`	A list of trees of class `phylo`, with identically labelled tips.
`Measure`	a function that calculates tree similarity or difference from quartet statuses. Default is `QuartetDivergence()`.

Value

a matrix specifying the distance between each tree and each other tree in the `trees`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

- Use splits (groups/clades defined by nodes or edges of the tree) instead of quartets as the unit of comparison: `SplitStatus()`.
- Generate distance metrics from quartet statuses: `SimilarityMetrics()`.

Other element-by-element comparisons: `CompareQuartets()`, `CompareQuartetsMulti()`, `CompareSplits()`, `PairSharedQuartetStatus()`, `QuartetState()`, `SharedQuartetStatus()`, `SplitStatus()`

Examples

```
data("sq_trees")
# Calculate the status of each quartet relative to the first entry in
# sq_trees
sq_status <- QuartetStatus(sq_trees)

# Calculate Estabrook et al's similarity measures:
SimilarityMetrics(sq_status)

# Compare trees that include a subset of the taxa 1..10
```

```
library("TreeTools", quietly = TRUE, warn.conflict = FALSE)
QuartetStatus(BalancedTree(1:5), BalancedTree(3:8), nTip = 10)

# If all taxa studied occur in `trees` or `cf`, set `nTip = TRUE`
QuartetStatus(BalancedTree(1:5), BalancedTree(3:10), nTip = TRUE)
```

`PlotQuartet` *Plot quartet on tree topologies*

Description

Draws a tree, highlighting the members of a specified quartet in colour.

Usage

```
PlotQuartet(tree, quartet, overwritePar = TRUE, caption = TRUE, ...)
```

Arguments

`tree`	A tree of class `phylo`, or a list of such trees. The first member of `tree` will be considered the "reference" tree.
`quartet`	A vector of four integers, corresponding to numbered leaves on the tree; or a character vector specifying the labels of four leaves.
`overwritePar`	Logical specifying whether to use existing `mfrow` and `mar` parameters from `par()` (`FALSE`), or to plot trees side-by-side in a new graphical device (`TRUE`).
`caption`	Logical specifying whether to annotate each plot to specify whether the quartet selected is in the same or a different state to the reference tree.
`...`	Additional parameters to send to `plot()`.

Value

`PlotQuartet()` returns `invisible()`, having plotted a tree in which the first two members of `quartet` are highlighted in orange, and the second two highlighted in blue.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other visualization: `QuartetPoints()`, `SymmetricDifferenceLineEnds()`, `VisualizeQuartets()`

Examples

```
data("sq_trees")

oPar <- par(mfrow = c(3, 6), mar = rep(0.5, 4))
PlotQuartet(sq_trees, c(2, 5, 3, 8), overwritePar = FALSE)
par(oPar)
```

`QuartetPoints` *Plot tree differences on ternary plots*

Description

Generate points to depict tree difference (in terms of resolution and accuracy) on a ternary plot, per Smith (2019).

Usage

```
QuartetPoints(trees, cf = trees[[1]])

SplitPoints(trees, cf = trees[[1]])

BipartitionPoints(trees, cf = trees[[1]])
```

Arguments

`trees`	A list of trees of class `phylo`, with identically labelled tips.
`cf`	Comparison tree of class `phylo`. If unspecified, each tree is compared to the first tree in `trees`.

Details

The ternary plot (produced using the **Ternary** package, Smith 2017) will depict the number of quartets or splits that are:

- resolved in the reference tree (`cf`), but neither present nor contradicted in each comparison tree (`trees`);
- resolved differently in the reference and the comparison tree;
- resolved in the same manner in the reference and comparison trees.

If the reference tree (`cf`) is taken to represent the best possible knowledge of the "true" topology, then polytomies in the reference tree represent uncertainty. If a tree in `trees` resolves relationships within this polytomy, it is not possible to establish (based only on the reference tree) whether this resolution is correct or erroneous. As such, extra resolution in `trees` that is neither corroborated nor contradicted by `cf` is ignored.

Value

A data frame listing the ternary coordinates of trees, based on the amount of information that they have in common with the comparison tree (which defaults to the first member of the list, if unspecified).

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Smith MR (2017). "Ternary: An R Package for Creating Ternary Plots." doi:10.5281/zenodo.1068997.

Smith MR (2019). "Bayesian and parsimony approaches reconstruct informative trees from simulated morphological datasets." *Biology Letters*, **15**(2), 20180632. doi:10.1098/rsbl.2018.0632.

See Also

Other visualization: `PlotQuartet()`, `SymmetricDifferenceLineEnds()`, `VisualizeQuartets()`

Examples

```
library("Ternary")
data("sq_trees")

TernaryPlot(alab = "Unresolved", blab = "Contradicted", clab = "Consistent",
            point = "right")
TernaryLines(list(c(0, 2/3, 1/3), c(1, 0, 0)), col = "red", lty = "dotted")
TernaryText(QuartetPoints(sq_trees, cf = sq_trees$collapse_one), 1:15,
            col = Ternary::cbPalette8[2], cex = 0.8)
TernaryText(SplitPoints(sq_trees, cf = sq_trees$collapse_one), 1:15,
            col = Ternary::cbPalette8[3], cex = 0.8)
legend("bottomright", c("Quartets", "Splits"), bty = "n", pch = 1, cex = 0.8,
       col = Ternary::cbPalette8[2:3])
```

QuartetState *Quartet State(s)*

Description

Report the status of the specified quartet(s) in given trees or lists of splits (Estabrook et al. 1985).

Usage

```
QuartetState(tips, bips, splits = bips, asRaw = FALSE)

QuartetStates(splits, asRaw = FALSE)

## S3 method for class 'Splits'
QuartetStates(splits, asRaw = FALSE)

## S3 method for class 'list'
QuartetStates(splits, asRaw = FALSE)

## S3 method for class 'multiPhylo'
QuartetStates(splits, asRaw = FALSE)
```

Arguments

tips	A four-element array listing a quartet of leaves, either by their number (if class numeric) or their name (if class character).
bips	Deprecated; included for compatibility with v1.0.2 and below.
splits	An object, such as a tree of class phylo, that can be induced to a Splits object using as.Splits.
asRaw	Logical specifying whether return format should be raw, which uses less memory and can be processed faster than integer type. Default is currently set to FALSE for backwards compatibility; suggest overriding to TRUE.

Details

One of the three possible four-leaf trees will be consistent with any set of splits generated from a fully resolved tree. If the leaves are numbered 1 to 4, this tree can be identified by naming the leaf most closely related to leaf 4. If a set of splits is generated from a tree that contains polytomies, it is possible that all three four-leaf trees are consistent with the set of splits.

Value

`QuartetState()` returns `0` if the relationships of the four leaves are not constrained by the provided splits, or the index of the closest relative to `tips[4]`, otherwise.

`QuartetStates()` returns a raw vector listing the status of each quartet of leaves (in the order listed by `AllQuartets()`) in turn, or if multiple trees are provided, a matrix in which each row corresponds to such a vector.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.

See Also

Compare quartet states between trees (slowly) using `CompareQuartets()` and `CompareQuartetsMulti()`.

Other element-by-element comparisons: `CompareQuartets()`, `CompareQuartetsMulti()`, `CompareSplits()`, `PairSharedQuartetStatus()`, `PairwiseQuartets()`, `SharedQuartetStatus()`, `SplitStatus()`

Examples

```
trees <- list(TreeTools::BalancedTree(6),
              TreeTools::PectinateTree(6))

trees[[3]] <- TreeTools::CollapseNode(trees[[2]], 9:10)

QuartetState(c(1, 3, 4, 6), trees[[2]])
QuartetState(1:4, trees[[1]]) == QuartetState(1:4, trees[[2]])
QuartetState(c(1, 3, 4, 6), trees[[3]])

QuartetStates(trees[[2]])
QuartetStates(trees[[3]])

CompareQuartets(QuartetStates(trees[[2]]), QuartetStates(trees[[3]]))
CompareQuartetsMulti(trees[[1]], trees[2:3])
```

ResolvedQuartets *Count resolved quartets*

Description

Counts how many quartets are resolved or unresolved in a given tree, following Brodal et al. (2013).

Usage

```
ResolvedQuartets(tree, countTriplets = FALSE)

ResolvedTriplets(tree)
```

Arguments

`tree`	A tree of class `phylo`.
`countTriplets`	Logical; if `TRUE`, the function will return the number of triplets instead of the number of quartets.

Details

Trees with more than 477 leaves risk encountering integer overflow errors, as the number of quartets is larger than can be stored in R's signed 32-bit integer representation. If warnings are thrown, check subsequent calculations for errors.

Value

`ResolvedQuartets()` returns a vector of length two, listing the number of quartets (or triplets) that are `[1]` resolved; `[2]` unresolved in the specified tree.

Functions

- `ResolvedTriplets()`: Convenience function to calculate the number of resolved/unresolved triplets.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Brodal GS, Fagerberg R, Mailund T, Pedersen CNS, Sand A (2013). "Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree." *SODA '13 Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1814–1832. doi:10.1137/1.9781611973105.130.

See Also

Other enumeration: `AllQuartets()`, `CompareQuartets()`, `CompareQuartetsMulti()`

Examples

```
data(sq_trees)

ResolvedTriplets(sq_trees$collapse_some)
# Equivalent to:
ResolvedQuartets(sq_trees$collapse_some, countTriplets = TRUE)

vapply(sq_trees, ResolvedQuartets, integer(2))
```

SharedQuartetStatus *Status of quartets*

Description

Determines the number of quartets that are consistent within pairs of trees.

Usage

```
SharedQuartetStatus(trees, cf = trees[[1]])

QuartetStatus(trees, cf = trees[[1]], nTip = NULL)

ManyToManyQuartetAgreement(trees, nTip = NULL)

TwoListQuartetAgreement(trees1, trees2)

SingleTreeQuartetAgreement(trees, comparison)
```

Arguments

`trees`	A list of trees of class `phylo`, with identically labelled tips.
`cf`	Comparison tree of class `phylo`. If unspecified, each tree is compared to the first tree in `trees`.
`nTip`	Integer specifying number of tips that could have occurred in `trees`. Useful if comparing trees from different data sources that contain non-overlapping tips. If `NULL`, the default, then trees are assumed to contain the same tips. If `TRUE`, then a vector is generated automatically by counting all unique tip labels found in `trees` or `cf`.
`trees1`, `trees2`	List or `multiPhylo` objects containing trees of class `phylo`.
`comparison`	A tree of class `phylo` against which to compare `trees`.

Details

Given a list of trees, returns the number of quartet statements (Estabrook et al. 1985) present in the reference tree (the first entry in `trees`, if `cf` is not specified) that are also present in each other tree. A random pair of fully resolved trees is expected to share `choose(n_tip, 4) / 3` quartets.

If trees do not bear the same number of tips, `SharedQuartetStatus()` will consider only the quartets that include taxa common to both trees.

From this information it is possible to calculate how many of all possible quartets occur in one tree or the other, though there is not yet a function calculating this; let us know if you would appreciate this functionality.

The status of each quartet is calculated using the algorithms of Brodal et al. (2013) and Holt et al. (2014), implemented in the tqdist C library (Sand et al. 2014).

Value

`QuartetStatus()` returns a two dimensional array. Rows correspond to the input trees; the first row will report a perfect match if the first tree is specified as the comparison tree (or if `cf` is not specified). Columns list the status of each quartet:

N The total number of quartet *statements* for two trees of n leaves, i.e. $2\ Q$.

Q The total number of quartets for n leaves.

s The number of quartets that are resolved identically in both trees.

d The number of quartets that are resolved differently in each tree.

r1 The number of quartets that are resolved in tree 1, but not in tree 2.

r2 The number of quartets that are resolved in tree 2, but not in tree 1.

u The number of quartets that are unresolved in both trees.

`ManyToManyQuartetAgreement()` returns a three-dimensional array listing, for each pair of trees in turn, the number of quartets in each category.

`TwoListQuartetAgreement()` returns a three-dimensional array listing, for each pair of trees in turn, the number of quartets in each category.

`SingleTreeQuartetAgreement()` returns a two-dimensional array listing, for tree in `trees`, the total number of quartets and the number of quartets in each category. The `comparison` tree is treated as `tree2`.

Functions

- `SharedQuartetStatus()`: Reports split statistics obtained after removing all tips that do not occur in both trees being compared.
- `ManyToManyQuartetAgreement()`: Agreement of each quartet, comparing each pair of trees in a list.
- `TwoListQuartetAgreement()`: Agreement of each quartet in trees in `trees1` with each quartet in trees in `trees2`.
- `SingleTreeQuartetAgreement()`: Agreement of each quartet in trees in a list with the quartets in a comparison tree.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Brodal GS, Fagerberg R, Mailund T, Pedersen CNS, Sand A (2013). "Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree." *SODA '13 Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1814–1832. doi:10.1137/1.9781611973105.130.

Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.

Holt MK, Johansen J, Brodal GS (2014). "On the scalability of computing triplet and quartet distances." In *Proceedings of 16th Workshop on Algorithm Engineering and Experiments (ALENEX) Portland, Oregon, USA*.

Sand A, Holt MK, Johansen J, Brodal GS, Mailund T, Pedersen CNS (2014). "tqDist: a library for computing the quartet and triplet distances between binary or general trees." *Bioinformatics*, **30**(14), 2079–2080. ISSN 1460-2059. doi:10.1093/bioinformatics/btu157.

See Also

- Use splits (groups/clades defined by nodes or edges of the tree) instead of quartets as the unit of comparison: SplitStatus().
- Generate distance metrics from quartet statuses: SimilarityMetrics(), [PairwiseQuartets()].

Other element-by-element comparisons: CompareQuartets(), CompareQuartetsMulti(), CompareSplits(), PairSharedQuartetStatus(), PairwiseQuartets(), QuartetState(), SplitStatus()

Examples

```
data("sq_trees")
# Calculate the status of each quartet relative to the first entry in
# sq_trees
sq_status <- QuartetStatus(sq_trees)

# Calculate the status of each quartet relative to a given tree
two_moved <- sq_trees[5:7]
sq_status <- QuartetStatus(two_moved, sq_trees$ref_tree)

# Calculate Estabrook et al's similarity measures:
SimilarityMetrics(sq_status)

# Compare trees that include a subset of the taxa 1..10
library("TreeTools", quietly = TRUE, warn.conflict = FALSE)
QuartetStatus(BalancedTree(1:5), BalancedTree(3:8), nTip = 10)

# If all taxa studied occur in `trees` or `cf`, set `nTip = TRUE`
QuartetStatus(BalancedTree(1:5), BalancedTree(3:10), nTip = TRUE)

# Calculate Quartet Divergence between each tree and each other tree in a
# list
QuartetDivergence(ManyToManyQuartetAgreement(two_moved))
# Calculate Quartet Divergence between each tree in one list and each
# tree in another
QuartetDivergence(TwoListQuartetAgreement(sq_trees[1:3], sq_trees[10:13]))
```

SimilarityMetrics *Tree similarity measures*

Description

Measure tree similarity or difference.

Usage

```
SimilarityMetrics(elementStatus, similarity = TRUE)

DoNotConflict(elementStatus, similarity = TRUE)
```

```
ExplicitlyAgree(elementStatus, similarity = TRUE)

StrictJointAssertions(elementStatus, similarity = TRUE)

SemiStrictJointAssertions(elementStatus, similarity = TRUE)

SymmetricDifference(elementStatus, similarity = TRUE)

RawSymmetricDifference(elementStatus, similarity = FALSE)

RobinsonFoulds(elementStatus, similarity = FALSE)

MarczewskiSteinhaus(elementStatus, similarity = TRUE)

SteelPenny(elementStatus, similarity = TRUE)

QuartetDivergence(elementStatus, similarity = TRUE)

SimilarityToReference(elementStatus, similarity = TRUE, normalize = FALSE)
```

Arguments

`elementStatus`	Two-dimensional integer array, with rows corresponding to counts of matching quartets or partitions for each tree, and columns named according to the output of `QuartetStatus()` or `SplitStatus()`.
`similarity`	Logical specifying whether to calculate the similarity or dissimilarity.
`normalize`	Logical; if `TRUE`, a random or star tree has expected similarity 0 (or difference 1), and the maximum possible score is one. If `FALSE`, zero similarity corresponds to all quartets contradicted, whereas one corresponds to all quartets correctly resolved – which will be unattainable if the reference tree contains polytomies.

Details

Estabrook et al. (1985) (table 2) define four similarity metrics in terms of the total number of quartets (N, their Q), the number of quartets resolved in the same manner in two trees (s), the number resolved differently in both trees (d), the number resolved in tree 1 or 2 but unresolved in the other tree ($r1$, $r2$), and the number that are unresolved in both trees (u).

The similarity metrics are then given as below. The dissimilarity metrics are their complement (i.e. 1 - *similarity*), and can be calculated algebraically using the identity $N = s + d + r1 + r2 + u$.

Although defined using quartets, analogous values can be calculated using partitions – though for a number of reasons, quartets may offer a more meaningful measure of the amount of information shared by two trees (Smith 2020).

- Do Not Conflict (DC): $(s + r1 + r2 + u) / N$
- Explicitly Agree (EA): s / N
- Strict Joint Assertions (SJA): $s / (s + d)$
- SemiStrict Joint Assertions (SSJA): $s / (s + d + u)$

(The numerator of the SemiStrict Joint Assertions similarity metric is given in Estabrook et al. (1985) table 2 as $s + d$, but this is understood, with reference to their text, to be a typographic error.)

Steel and Penny (1993) propose a further metric, which they denote d_Q_, which this package calculates using the function `SteelPenny()`:

- Steel & Penny's quartet metric (dQ): $(s + u) / N$

Another take on tree similarity is to consider the symmetric difference: that is, the number of partitions or quartets present in one tree that do not appear in the other, originally used to measure tree similarity by Robinson and Foulds (1981). (Note that, given the familiarity of the Robinson–Foulds distance metric, this quantity is be default expressed as a difference rather than a similarity.)

- Raw symmetric difference (RF): $d1 + d2 + r1 + r2$

A pair of trees will have a high symmetric difference if they are well-resolved but disagree on many relationships; or if they agree on most relationships but are poorly resolved. As such, it is essential to contextualize the symmetric difference by appropriate normalization (Smith 2019). Multiple approaches to normalization have been proposed:

The total number of resolved quartets or partitions present in both trees (Day 1986):

- Symmetric Difference (SD): $(2\ d + r1 + r2) / (2\ d + 2\ s + r1 + r2)$

The total distinctly resolved quartets or partitions (Marczewski and Steinhaus 1958; Day 1986):

- Marczewski-Steinhaus (MS): $(2\ d + r1 + r2) / (2\ d + s + r1 + r2)$

The maximum number of quartets or partitions that could have been resolved, given the number of tips (Smith 2019):

- Symmetric Divergence: $(d + d + r1 + r2) / N$

Finally, in cases where a reconstructed tree `r1` is being compared to a reference tree `r2` taken to represent "true" relationships, a symmetric difference is not desired. In such settings, the desired score is the expectation that a given quartet's resolution in the reconstructed tree is "correct", given by Asher and Smith (2022):

- Similarity to Reference (S2R): $(s + (r1 + r2 + u) / 3) / Q$

This may optionally be normalized with reference to the maximum possible similarity, $(s + d + r2 + (r1 + u) / 3) / Q$, subtracting 1/3 (the probability of matching at random) from both the S2R score and maximum possible score before dividing; then, a tree scores zero if it is as different from the true tree as a random or fully unresolved tree, and one if it is as "true" as can be known.

Value

`SimilarityMetrics()` returns a named two-dimensional array in which each row corresponds to an input tree, and each column corresponds to one of the listed measures.

`DoNotConflict()` and others return a named vector describing the requested similarity (or difference) between the trees.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Asher R, Smith MR (2022). "Phylogenetic signal and bias in paleontology." *Systematic Biology*, **71**(4), 986–1008. doi:10.1093/sysbio/syab072.

Day WH (1986). "Analysis of quartet dissimilarity measures between undirected phylogenetic trees." *Systematic Biology*, **35**(3), 325–333. doi:10.1093/sysbio/35.3.325.

Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.

Marczewski E, Steinhaus H (1958). "On a certain distance of sets and the corresponding distance of functions." *Colloquium Mathematicae*, **6**(1), 319–327.

Robinson DF, Foulds LR (1981). "Comparison of phylogenetic trees." *Mathematical Biosciences*, **53**(1-2), 131–147. doi:10.1016/00255564(81)900432.

Smith MR (2019). "Bayesian and parsimony approaches reconstruct informative trees from simulated morphological datasets." *Biology Letters*, **15**(2), 20180632. doi:10.1098/rsbl.2018.0632.

Smith MR (2020). "Information theoretic Generalized Robinson-Foulds metrics for comparing phylogenetic trees." *Bioinformatics*, **36**(20), 5007–5013. doi:10.1093/bioinformatics/btaa614.

Steel MA, Penny D (1993). "Distributions of tree comparison metrics—some new results." *Systematic Biology*, **42**(2), 126–141. doi:10.1093/sysbio/42.2.126. https://www.math.canterbury.ac.nz/~m.steel/Non_UC/files/research/distributions.pdf.

See Also

- Calculate status of each quartet – the raw material from which the Estabrook *et al.* metrics are calculated – with `QuartetStatus()`:
- Equivalent metrics for bipartition splits: `SplitStatus()`, `CompareSplits()`

Examples

```
data("sq_trees")

sq_status <- QuartetStatus(sq_trees)
SimilarityMetrics(sq_status)
QuartetDivergence(sq_status, similarity = FALSE)

library("TreeTools", quietly = TRUE, warn.conflict = FALSE)
set.seed(0)
reference <- CollapseNode(as.phylo(101, 10), 16:18)
trees <- c(
  reference = reference,
  binaryRef = MakeTreeBinary(reference),
  balanced = BalancedTree(reference),
  pectinate = PectinateTree(reference),
```

```
  star = StarTree(reference),
  random = RandomTree(reference),
  random2 = RandomTree(reference)
)
elementStatus <- QuartetStatus(trees, reference)
SimilarityToReference(elementStatus)
SimilarityToReference(elementStatus, normalize = TRUE)
```

`SplitStatus` *Matching partitions*

Description

Calculates how many of the partitions present in tree 1 are also present in tree 2 (`s`), how many of the partitions in tree 1 are absent in tree 2 (`d1`), and how many of the partitions in tree 2 are absent in tree 1 (`d2`). The Robinson-Foulds (symmetric partition) distance is the sum of the latter two quantities, i.e. `d1 + d2`.

Usage

```
SplitStatus(trees, cf = trees[[1]])

SharedSplitStatus(trees, cf)
```

Arguments

`trees`	A list of trees of class `phylo`, with identically labelled tips.
`cf`	Comparison tree of class `phylo`. If unspecified, each tree is compared to the first tree in `trees`.

Value

Returns a two dimensional array. Rows correspond to the input trees, and are named if names were present. Columns report:

N: The total number of partitions present in the two trees, i.e. *P1 + P2*.

P1: The number of partitions present in tree 1.

P2: The number of partitions present in tree 2.

s: The number of partitions present in both trees.

d1: The number of partitions present in tree 1, but contradicted by tree 2.

d2: The number of partitions present in tree 2, but contradicted by tree 1.

r1: The number of partitions present in tree 1, and neither present nor contradicted in tree 2.

r2: The number of partitions present in tree 2, and neither present nor contradicted in tree 1.

Functions

- `SharedSplitStatus()`: Reports split statistics obtained after removing all tips that do not occur in both trees being compared.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

- Robinson DF, Foulds LR (1981). "Comparison of phylogenetic trees." *Mathematical Biosciences*, **53**(1-2), 131–147. doi:10.1016/00255564(81)900432.
- Penny D, Hendy MD (1985). "The use of tree comparison metrics." *Systematic Zoology*, **34**(1), 75–82. doi:10.2307/2413347.

See Also

Other element-by-element comparisons: `CompareQuartets()`, `CompareQuartetsMulti()`, `CompareSplits()`, `PairSharedQuartetStatus()`, `PairwiseQuartets()`, `QuartetState()`, `SharedQuartetStatus()`

Examples

```
data("sq_trees")

# Calculate the status of each quartet
splitStatuses <- SplitStatus(sq_trees)

# Calculate the raw symmetric difference (i.e. Robinson-Foulds distance)
RawSymmetricDifference(splitStatuses)

# Normalize the Robinson Foulds distance by dividing by the number of
# splits present in the two trees:
RawSymmetricDifference(splitStatuses) / splitStatuses[, "N"]

# Normalize the Robinson Foulds distance by dividing by the total number of
# splits that it is possible to resolve for `n` tips:
nTip <- length(sq_trees[[1]]$tip.label)
nPartitions <- 2 * (nTip - 3L) # Does not include the nTip partitions that
                               # comprise but a single tip
RawSymmetricDifference(splitStatuses) / nPartitions
```

sq_trees *Eighteen example trees*

Description

A list of class multiPhylo containing phylogenetic trees:

ref_tree A reference tree, bearing tips labelled 1 to 11.

move_one_near Tip 1 has been moved a short distance.

move_one_mid Tip 1 has been moved further.

move_one_far Tip 1 has been moved further still.

move_two_near Tips 10 & 11 have been moved a short distance.

move_two_mid Tips 10 & 11 have been moved further.

move_two_far Tips 10 & 11 have been moved further still.

collapse_one One node has been collapsed into a polytomy.

collapse_some Several nodes have been collapsed.

m1mid_col1 Tree move_one_mid with one node collapsed.

m1mid_colsome Tree move_one_mid with several nodes collapsed.

m2mid_col1 Tree move_two_mid with one node collapsed.

m2mid_colsome Tree move_two_mid with several nodes collapsed.

opposite_tree A tree that shares fewer quartets with ref_tree than expected by chance.

caterpillar A pectinate "caterpillar" tree.

top_and_tail Tree caterpillar, with its outermost taxa swapped such that it shares no partitions with caterpillar.

anti_pectinate A random tree that shares no partitions with caterpillar.

random_tree A random tree.

Usage

```
sq_trees
```

Format

An object of class multiPhylo of length 18.

`SymmetricDifferenceLineEnds` *Plot contours of equal symmetric difference on a ternary plot*

Description

Assumes that tree 1 is perfectly resolved, but that the resolution of tree 2 can vary.

Usage

```
SymmetricDifferenceLineEnds(nsd)

SymmetricDifferenceLines(nsd, ...)
```

Arguments

`nsd`	Vector specifying normalized symmetric differences to plot.
`...`	Further parameters to pass to `TernaryLines()`.

Value

Returns a matrix of dim `(length(nsd), 6)`, with columns named `r2a`, `da`, `sa`, `r2b`, `db` and `sb`. Lines from `a` to `b` in each row connect points of equal symmetric difference.

Functions

- `SymmetricDifferenceLines()`: Plot the lines onto the active ternary plot.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other visualization: `PlotQuartet()`, `QuartetPoints()`, `VisualizeQuartets()`

`TQDist` *Wrapper for tqDist*

Description

`TQDist()` and `TQAE()` are convenience functions that writes a list of trees to text files that can be processed by the C implementation of tqDist (Sand et al. 2014). tqDist is then called, and the temporary file is deleted when analysis is complete.

Usage

```
TQDist(trees)

TQAE(trees)
```

Arguments

`trees`	List of phylogenetic trees, of class `list` or `multiPhylo`.

Details

Quartets can be resolved in one of five ways, which Brodal et al. (2013) and Holt et al. (2014) distinguish using the letters A-E, and Estabrook et al. (1985) refer to as:

A *s* = resolved the **s**ame in both trees;

B *d* = resolved **d**ifferently in both trees;

C *r1* = **r**esolved only in tree **1**;

D *r2* = **r**esolved only in tree **2** (the comparison tree);

E *u* = **u**nresolved in both trees.

Value

`TQDist()` returns the quartet distance between each pair of trees.

`TQAE()` returns the number of resolved quartets in agreement between each pair of trees ("A" in Brodal *et al*. 2013) and the number of quartets that are unresolved in both trees ("E" in Brodal *et al*. 2013).

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

References

Brodal GS, Fagerberg R, Mailund T, Pedersen CNS, Sand A (2013). "Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree." *SODA '13 Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1814–1832. doi:10.1137/1.9781611973105.130.

Estabrook GF, McMorris FR, Meacham CA (1985). "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units." *Systematic Zoology*, **34**(2), 193–200. doi:10.2307/2413326.

Holt MK, Johansen J, Brodal GS (2014). "On the scalability of computing triplet and quartet distances." In *Proceedings of 16th Workshop on Algorithm Engineering and Experiments (ALENEX) Portland, Oregon, USA*.

Sand A, Holt MK, Johansen J, Brodal GS, Mailund T, Pedersen CNS (2014). "tqDist: a library for computing the quartet and triplet distances between binary or general trees." *Bioinformatics*, **30**(14), 2079–2080. ISSN 1460-2059. doi:10.1093/bioinformatics/btu157.

See Also

`CompareQuartets()`, `QuartetStatus()`

Other TQDist functions: `TripletDistance.phylo()`

`TripletDistance.phylo` *Direct entry points to "tqDist" functions*

Description

Wrappers for functions in "tqDist", which calculate triplet and quartet distances between pairs of trees.

Usage

```
## S3 method for class 'phylo'
TripletDistance(tree1, tree2 = NULL)

## S3 method for class 'list'
TripletDistance(tree1, tree2 = NULL)

## S3 method for class 'multiPhylo'
TripletDistance(tree1, tree2 = NULL)

QuartetDistance(file1, file2)

QuartetAgreement(file1, file2)
```

```
PairsQuartetDistance(file1, file2)

OneToManyQuartetAgreement(file1, file2)

AllPairsQuartetDistance(file)

AllPairsQuartetAgreement(file)

TripletDistance(tree1, tree2 = NULL)

## S3 method for class 'character'
TripletDistance(tree1, tree2 = NULL)

PairsTripletDistance(file1, file2)

AllPairsTripletDistance(file)
```

Arguments

`tree1`, `tree2`	Arguments to `TripletDistance`: file paths containing Newick trees, phylogenetic trees of class `phylo`, or lists/`multiPhylo` of trees. If `tree1` is a list and `tree2 = NULL`, returns a distance matrix of all pairwise distances.
`file`, `file1`, `file2`	Paths to files containing a tree or trees in Newick format, possibly created using `TQFile()`.

Value

`...Distance()` functions return the distance between the requested trees.

`...Agreement()` functions return the number of triplets or quartets that are:

- `A`, resolved in the same fashion in both trees;
- `E`, unresolved in both trees.

Comparing a tree against itself yields the totals (`A+B+C`) and (`D+E`) referred to by Brodal et al. (2013) and Holt et al. (2014).

Functions

- `TripletDistance(phylo)`: Triplet distance between two trees of class `phylo`. Uses the CPDT algorithm (Jansson and Rajaby 2017).
- `TripletDistance(list)`: Triplet distance between each pair of trees in a list. If `tree2` is provided, returns pairwise distances between corresponding trees in `tree1` and `tree2`. If `tree2 = NULL`, returns a distance matrix of all pairwise distances within `tree1`.
- `TripletDistance(multiPhylo)`: Triplet distance between each pair of trees in a `multiPhylo` object.
- `QuartetDistance()`: Returns the quartet distance between the tree. in `file1` and the tree in `file2`.

- `QuartetAgreement()`: Returns a vector of length two, listing [1] the number of resolved quartets that agree (`A`); [2] the number of quartets that are unresolved in both trees (`E`). See Brodal et al. (2013).
- `PairsQuartetDistance()`: Quartet distance between the tree on each line of `file1` and the tree on the corresponding line of `file2`.
- `OneToManyQuartetAgreement()`: Quartet distance between the tree in `file1` and the tree on each line of `file2`.
- `AllPairsQuartetDistance()`: Quartet distance between each tree listed in `file` and each other tree therein.
- `AllPairsQuartetAgreement()`: Quartet status for each pair of trees in `file`.
- `TripletDistance()`: Triplet distance between trees. `tree1` and `tree2` can be `phylo` objects, file paths to Newick tree files, or lists/`multiPhylo` objects. If `tree1` is a list and `tree2 = NULL`, computes all pairwise distances; if both are lists, computes distances between corresponding pairs. Uses the CPDT algorithm (Jansson and Rajaby 2017).
- `TripletDistance(character)`: Triplet distance between single trees read from Newick-format files `tree1` and `tree2`.
- `PairsTripletDistance()`: Triplet distance between the tree on each line of `file1` and the tree on the corresponding line of `file2`.
- `AllPairsTripletDistance()`: Triplet distance between each tree listed in `file` and each other tree therein.

Author(s)

- Quartet algorithms: Brodal et al. (2013); Holt et al. (2014). C implementation: Sand et al. (2014); modified for portability by Martin R. Smith.
- Triplet algorithm: Jansson and Rajaby (2017). C++ implementation by Ramesh Rajaby; R integration by Martin R. Smith.

References

Brodal GS, Fagerberg R, Mailund T, Pedersen CNS, Sand A (2013). "Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree." *SODA '13 Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1814–1832. doi:10.1137/1.9781611973105.130.

Holt MK, Johansen J, Brodal GS (2014). "On the scalability of computing triplet and quartet distances." In *Proceedings of 16th Workshop on Algorithm Engineering and Experiments (ALENEX) Portland, Oregon, USA*.

Jansson J, Rajaby R (2017). "A More Practical Algorithm for the Rooted Triplet Distance." *Journal of Computational Biology*, **24**(2), 106–126. doi:10.1089/cmb.2016.0185.

See Also

- `QuartetStatus()` takes trees, rather than files, as input.
- `TQFile()` creates a temporary file containing specified trees.

Other TQDist functions: `TQDist()`

`VisualizeQuartets` *Visualize quartet difference on trees, by split*

Description

Visualize quartet difference on trees, by split

Usage

```
VisualizeQuartets(
  tree1,
  tree2,
  style = "pie",
  setPar = TRUE,
  precision = 3L,
  Plot = plot.phylo,
  scale = 1L,
  spectrum = hcl.colors(101, palette = "viridis"),
  legend = TRUE,
  ...
)
```

Arguments

`tree1`, `tree2`	Trees of class `phylo`, with identical leaf labels.
`style`	Character string specifying split labels with an unambiguous abbreviation of: • `label`: Label stating proportion of resolved quartets in agreement, coloured accordingly; • `pie`: Pie chart showing proportion of quartets in agreement, sized according to number of quartets influenced by each split; • `bar`: Bar showing proportion of quartets in agreement, labelled; • `size`: Circle coloured according to proportion of quartets in agreement, with area corresponding to number of quartet statements associated with split.
`setPar`	Logical specifying whether graphical parameters should be set to display trees side by side.
`precision`	Integer specifying number of significant figures to display when reporting matching scores.
`Plot`	Function to use to plot trees.
`scale`	Numeric, enlargement factor for split labels.
`spectrum`	101-element vector specifying a range of colours by which to colour matches.
`legend`	Logical specifying whether to display simple legend.
`...`	Additional parameters to send to `Plot()`.

Value

`VisualizeQuartets()` invisibly returns a list with two elements, named `tree1` and `tree2`, containing a matrix. Each row corresponds to a split within that tree; columns correspond to:

node The internal numbering of the node corresponding to each split, as displayed by `ape::nodelabels()`

N, Q, s, d, r1, r2, u The status of each quartet relative to that split, as documented in `QuartetStatus()`

res The number of quartets resolved by that split, i.e. `s + d`

same The proportion of quartets resolved by that node that are resolved in the same manner in the other tree; i.e. `s / s + d`

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

See Also

Other visualization: `PlotQuartet()`, `QuartetPoints()`, `SymmetricDifferenceLineEnds()`

Examples

```
library("TreeTools", quietly = TRUE)
# Simple plot
VisualizeQuartets(BalancedTree(10), CollapseNode(PectinateTree(10), 19),
                  style = "label")

# Plot with custom graphical parameters
origPar <- par(mfrow = c(2, 2))
VisualizeQuartets(BalancedTree(10), CollapseNode(PectinateTree(10), 19),
                  setPar = FALSE)
VisualizeQuartets(BalancedTree(10), CollapseNode(PectinateTree(10), 19),
                  style = "bar", legend = FALSE, setPar = FALSE)

# Circle size denotes similarity
par(mfrow = c(2, 1), mar = rep(0.1, 4))
vq <- VisualizeQuartets(
  tree1 = BalancedTree(20),
  tree2 = CollapseNode(PectinateTree(20), 29:33),
  style = "size", scale = 2,
  setPar = FALSE # necessary for node labels to align
)
# Manually add custom node labels
percentSame <- paste(round(vq[["tree2"]][, "same"] * 100, 1), "%")
nodelabels(percentSame, vq[["tree2"]][, "node"],
           frame = "n", bg = NA, # No frame or background
           cex = 0.8, # character expansion / font size
           adj = 0.5 # align label
           )

# restore original graphical parameters
par(origPar)
```

Index

∗ **TQDist functions**
  TQDist, 25
  TripletDistance.phylo, 26
∗ **Tree distances**
  TripletDistance.phylo, 26
∗ **datasets**
  sq_trees, 23
∗ **element-by-element comparisons**
  CompareQuartets, 4
  CompareQuartetsMulti, 5
  CompareSplits, 6
  PairwiseQuartets, 8
  QuartetState, 11
  SharedQuartetStatus, 14
  SplitStatus, 21
∗ **enumeration**
  AllQuartets, 3
  CompareQuartets, 4
  CompareQuartetsMulti, 5
  ResolvedQuartets, 13
∗ **tree comparison**
  SimilarityMetrics, 17
∗ **visualization**
  PlotQuartet, 9
  QuartetPoints, 10
  SymmetricDifferenceLineEnds, 24
  VisualizeQuartets, 29

AllPairsQuartetAgreement *(*TripletDistance.phylo*)*, 26
AllPairsQuartetDistance *(*TripletDistance.phylo*)*, 26
AllPairsTripletDistance *(*TripletDistance.phylo*)*, 26
AllQuartets, 3
AllQuartets(), *4*, *6*, *12*, *14*
as.Splits, *6*, *12*

BipartitionPoints *(*QuartetPoints*)*, 10
BipartitionStatus *(*SplitStatus*)*, 21

combn, *3*
CompareBipartitions *(*CompareSplits*)*, 6
CompareQuartets, 4
CompareQuartets(), *3*, *6–8*, *13*, *14*, *17*, *22*, *26*
CompareQuartetsMulti, 5
CompareQuartetsMulti(), *3*, *4*, *7*, *8*, *13*, *14*, *17*, *22*
CompareSplits, 6
CompareSplits(), *4*, *6*, *8*, *13*, *17*, *20*, *22*

Distances *(*TripletDistance.phylo*)*, 26
DoNotConflict *(*SimilarityMetrics*)*, 17

ExplicitlyAgree *(*SimilarityMetrics*)*, 17

ManyToManyQuartetAgreement *(*SharedQuartetStatus*)*, 14
MarczewskiSteinhaus *(*SimilarityMetrics*)*, 17
multiPhylo, *23*, *25*

OneToManyQuartetAgreement *(*TripletDistance.phylo*)*, 26

PairSharedQuartetStatus(), *4*, *6–8*, *13*, *17*, *22*
PairsQuartetDistance *(*TripletDistance.phylo*)*, 26
PairsTripletDistance *(*TripletDistance.phylo*)*, 26
PairwiseQuartets, 8
PairwiseQuartets(), *4*, *6*, *7*, *13*, *17*, *22*
par, *9*
phylo, *8–10*, *13*, *15*, *21*
plot, *9*
PlotQuartet, 9
PlotQuartet(), *11*, *24*, *30*

QuartetAgreement *(*TripletDistance.phylo*)*, 26

QuartetDistance
(TripletDistance.phylo), 26
QuartetDivergence (SimilarityMetrics), 17
QuartetDivergence(), *8*
QuartetPoints, 10
QuartetPoints(), *9*, *24*, *30*
QuartetState, 11
QuartetState(), *4*, *6–8*, *17*, *22*
QuartetStates (QuartetState), 11
QuartetStates(), *3*, *4*
QuartetStatus (SharedQuartetStatus), 14
QuartetStatus(), *4*, *18*, *20*, *26*, *28*, *30*

RawSymmetricDifference
(SimilarityMetrics), 17
ResolvedQuartets, 13
ResolvedQuartets(), *3*, *4*, *6*
ResolvedTriplets (ResolvedQuartets), 13
RobinsonFoulds (SimilarityMetrics), 17

SemiStrictJointAssertions
(SimilarityMetrics), 17
SharedBipartitionStatus (SplitStatus), 21
SharedQuartetStatus, 14
SharedQuartetStatus(), *4*, *6–8*, *13*, *22*
SharedSplitStatus (SplitStatus), 21
SimilarityMetrics, 17
SimilarityMetrics(), *8*, *17*
SimilarityToReference
(SimilarityMetrics), 17
SingleTreeQuartetAgreement
(SharedQuartetStatus), 14
SplitPoints (QuartetPoints), 10
SplitStatus, 21
SplitStatus(), *4*, *6–8*, *13*, *17*, *18*, *20*
sq_trees, 23
SteelPenny (SimilarityMetrics), 17
StrictJointAssertions
(SimilarityMetrics), 17
SymmetricDifference
(SimilarityMetrics), 17
SymmetricDifferenceLineEnds, 24
SymmetricDifferenceLineEnds(), *9*, *11*, *30*
SymmetricDifferenceLines
(SymmetricDifferenceLineEnds), 24

TernaryLines, *24*
TQAE (TQDist), 25
TQDist, 25
TQDist(), *28*
TQFile(), *27*, *28*
TripletDistance
(TripletDistance.phylo), 26
TripletDistance.phylo, 26
TripletDistance.phylo(), *26*
TwoListQuartetAgreement
(SharedQuartetStatus), 14

VisualizeQuartets, 29
VisualizeQuartets(), *9*, *11*, *24*