Package 'Rogue'

September 13, 2021

Title Identify Rogue Taxa in Sets of Phylogenetic Trees

Version 2.0.0

License GPL (>= 3)

Description Rogue ('wildcard') taxa are leaves with uncertain phylogenetic position.
Their position may vary from tree to tree under inference methods that yield a tree set (e.g. bootstrapping, Bayesian tree searches, maximum parsimony).
The presence of rogue taxa in a tree set can potentially remove all information from a consensus tree. The information content of a consensus tree - a function of its resolution and branch support values - can often be increased by removing rogue taxa.
'Rogue' provides an explicitly information-theoretic approach to rogue detection (Smith, forthcoming), and an interface to 'RogueNaRok' (Aberer et al. 2013) <doi:10.1093/sysbio/sys078>.

URL https://github.com/ms609/Rogue/,
https://github.com/aberer/RogueNaRok/,
https://github.com/ms609/RogueNaRok/

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

Depends R (>= 3.5.0)

Imports ape (>= 5.0), cli (>= 3.0), fastmatch, grDevices, matrixStats, Rdpack (>= 0.7), Rfast, stats, TreeDist (>= 2.1.1), TreeTools (>= 1.5.0), utils,

Suggests testthat,

Config/Needs/github-actions callr, pkgbuild, rcmdcheck,

Config/Needs/coverage covr

Config/Needs/memcheck devtools

Config/Needs/metadata codemetar

Config/Needs/website pkgdown

RdMacros Rdpack

SystemRequirements C99

ByteCompile true

Encoding UTF-8

Language en-GB

RoxygenNote 7.1.1

NeedsCompilation yes

Author Martin R. Smith [aut, cre, cph]
(<https://orcid.org/0000-0001-5660-1727>),
Andre J. Aberer [aut, cph]

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

Repository CRAN

Date/Publication 2021-09-13 15:30:21 UTC

R topics documented:

`C_RogueNaRok` *Drop rogue taxa to generate a more informative consensus*

Description

`RogueTaxa()` finds wildcard leaves whose removal increases the resolution or branch support values of a consensus tree, using the relative bipartition, shared phylogenetic, or mutual clustering concepts of information.

Usage

```
C_RogueNaRok(
  bootTrees = "",
  runId = "tmp",
  treeFile = "",
  computeSupport = TRUE,
  dropsetSize = 1,
  excludeFile = "",
  workDir = "",
  labelPenalty = 0,
  mreOptimization = FALSE,
  threshold = 50
)

RogueTaxa(
```

```
  trees,
  info = c("spic", "scic", "fspic", "fscic", "rbic"),
  return = c("taxa", "tree"),
  bestTree = NULL,
  computeSupport = TRUE,
  dropsetSize = 1,
  neverDrop = character(0),
  labelPenalty = 0,
  mreOptimization = FALSE,
  threshold = 50,
  verbose = FALSE
)

QuickRogue(
  trees,
  info = "phylogenetic",
  log = TRUE,
  average = "median",
  deviation = "mad",
  neverDrop,
  fullSeq = FALSE
)
```

Arguments

`bootTrees`	Path to a file containing a collection of bootstrap trees.
`runId`	An identifier for this run, appended to output files.
`treeFile, bestTree`	If a single best-known tree (such as an ML or MP tree) is provided, RogueNaRok optimizes the bootstrap support in this best-known tree (still drawn from the bootstrap trees); the `threshold` parameter is ignored.
`computeSupport`	Logical: If `FALSE`, then instead of trying to maximize the support in the consensus tree, RogueNaRok will try to maximize the number of bipartitions in the final tree by pruning taxa.
`dropsetSize`	Integer specifying maximum size of dropset per iteration. If `dropsetSize == n`, then RogueNaRok will test in each iteration which tuple of `n` taxa increases the optimality criterion the most, pruning taxa accordingly. This improves the result, but run times will increase at least linearly.
`excludeFile`	Taxa in this file (one taxon per line) will not be considered for pruning.
`workDir`	Path to a working directory where output files are created.
`labelPenalty`	A weight factor to penalize for dropset size when `info = 'rbic'`. The higher the value, the more conservative the algorithm is in pruning taxa. The default value of `0` gives the RBIC; `1` gives Pattengale's criterion.
`threshold, mreOptimization`	A threshold or mode for the consensus tree that is optimized. Specify a value between 50 (majority rule consensus, the default) and 100 (strict consensus), or set

	`mreOptimization = TRUE` for the extended majority rule consensus. Note that rogue taxa identified with respect to different thresholds can vary substantially.
`trees`	List of trees to analyse.
`info`	Concept of information to employ; see details.
`return`	If `taxa`, returns the leaves identified as rogues; if `tree`, return a consensus tree omitting rogue taxa.
`neverDrop`	Tip labels that should not be dropped from the consensus.
`verbose`	Logical specifying whether to display output from RogueNaRok. If `FALSE`, output will be included as an attribute of the return value.
`log`	Logical specifying whether to log-transform distances when calculating leaf stability.
`average`	Character specifying whether to use `'mean'` or `'median'` tip distances to calculate leaf stability.
`deviation`	Character specifying whether to use `'sd'` or `'mad'` to calculate leaf stability.
`fullSeq`	Logical specifying whether to list all taxa (`TRUE`), or only those that improve information content when all are dropped (`FALSE`).

Details

The splitwise phylogenetic information content measure produces the best results (Smith 202X). It uses the splitwise information content as a shortcut, which involves double counting of some information (which may or may not be desirable). The same holds for the mutual clustering information measure; this measure is less obviously suited to the detection of rogues.

The "relative bipartition information criterion" (RBIC) is the sum of all support values divided by the maximum possible support in a fully bifurcating tree with the initial set of taxa. The relative bipartition information content approach employs the 'RogueNaRok' implementation (Aberer et al. 2013), which can handle large trees relatively quickly. The RBIC is is not strictly a measure of information and can produce undesirable results (Wilkinson and Crotti 2017).

`C_RogueNaRok()` directly interfaces the 'RogueNaRok' C implementation, with no input checking; be aware that invalid input will cause undefined behaviour and is likely to crash R.

Value

`C_RogueNaRok()` returns `0` if successful; `-1` on error.

`RogueTaxa()` returns a `data.frame`. Each row after the first, which describes the starting tree, describes a dropset operation. Columns describe:

- `num`: Sequential index of the drop operation
- `taxNum`: Numeric identifier of the dropped leaves
- `taxon`: Text identifier of dropped leaves
- `rawImprovement`: Improvement in score obtained by this operation
- `IC`: Information content of tree after dropping all leaves so far, by the measure indicated by `info`.

Functions

- `QuickRogue`: Shortcut to 'fast' heuristic, with option to return evaluation of all taxa using `fullSeq = TRUE`.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk), linking to RogueNaRok C library by Andre Aberer (<andre.aberer at googlemail.com>)

References

Aberer AJ, Krompass D, Stamatakis A (2013). "Pruning rogue taxa improves phylogenetic accuracy: an efficient algorithm and webservice." *Systematic Biology*, **62**(1), 162–166. doi: 10.1093/sysbio/sys078.

Smith MR (202X). "Improving consensus trees by detecting rogue taxa." *Forthcoming*.

Wilkinson M, Crotti M (2017). "Comments on detecting rogue taxa using RogueNaRok." *Systematics and Biodiversity*, **15**(4), 291–295. doi: 10.1080/14772000.2016.1252440.

Examples

```
bootTrees <- system.file('example/150.bs', package = 'Rogue')
tmpDir <- tempdir()
C_RogueNaRok(bootTrees, workDir = tmpDir)

# Results have been written to our temporary directory
oldwd <- setwd(tmpDir)
read.table('RogueNaRok_droppedRogues.tmp', header = TRUE)

# Delete temporary files
file.remove('RogueNaRok_droppedRogues.tmp')
file.remove('RogueNaRok_info.tmp')

setwd(oldwd)
library("TreeTools", warn.conflicts = FALSE)

trees <- list(read.tree(text = ("(a, (b, (c, (d, (e, (X1, X2))))));")),
              read.tree(text = ("((a, (X1, X2)), (b, (c, (d, e))));")))
RogueTaxa(trees, dropsetSize = 2)

trees <- list(
     read.tree(text = '((a, y), (b, (c, (z, ((d, e), (f, (g, x)))))));'),
     read.tree(text = '(a, (b, (c, (z, (((d, y), e), (f, (g, x)))))));'),
     read.tree(text = '(a, (b, ((c, z), ((d, (e, y)), ((f, x), g)))));'),
     read.tree(text = '(a, (b, ((c, z), ((d, (e, x)), (f, (g, y))))));'),
     read.tree(text = '(a, ((b, x), ((c, z), ((d, e), (f, (g, y))))));')
     )
cons <- consensus(trees, p = 0.5)
plot(cons)
```

```
LabelSplits(cons, SplitFrequency(cons, trees) / length(trees))
reduced <- RogueTaxa(trees, info = 'phylogenetic', ret = 'tree')
plot(reduced)
LabelSplits(reduced, SplitFrequency(reduced, trees) / length(trees))

QuickRogue(trees, fullSeq = TRUE)
```

`TipInstability` *Tip instability*

Description

`TipInstability()` calculates the instability of each leaf in a tree. Unstable leaves are likely to display roguish behaviour.

Usage

```
TipInstability(
  trees,
  log = TRUE,
  average = "mean",
  deviation = "sd",
  checkTips = TRUE
)

ColByStability(trees, log = TRUE, average = "mean", deviation = "sd")
```

Arguments

`trees`	List of trees to analyse.
`log`	Logical specifying whether to log-transform distances when calculating leaf stability.
`average`	Character specifying whether to use `'mean'` or `'median'` tip distances to calculate leaf stability.
`deviation`	Character specifying whether to use `'sd'` or `'mad'` to calculate leaf stability.
`checkTips`	Logical specifying whether to check that tips are numbered consistently.

Details

I define the *instability* of a pair of leaves as the median absolute divergence in the cophenetic distance (the number of edges in the shortest path between the leaves) across all trees, normalized against the mean cophenetic distance. The instability of a single leaf is the mean instability of all pairs that include that leaf; higher values characterise leaves whose position is more variable between trees.

Author(s)

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

See Also

Other tip instability functions: `TipVolatility()`

Examples

```
library("TreeTools", quietly = TRUE)
trees <- AddTipEverywhere(BalancedTree(8), 'Rogue')[3:6]
plot(consensus(trees), tip.col = ColByStability(trees))
instab <- TipInstability(trees, log = FALSE, ave = 'mean', dev = 'mad')
plot(ConsensusWithout(trees, names(instab[instab > 0.2])))
```

`TipVolatility` *Detect rogue taxa using phylogenetic information distance*

Description

Calculate the volatility of each tip: namely, the impact on the mean phylogenetic information distance between trees when that tip is removed.

Usage

```
TipVolatility(trees)
```

Arguments

`trees`	List of trees to analyse.

Value

`TipVolatility()` returns a named vector listing the volatility index calculated for each leaf.

References

Aberer AJ, Krompass D, Stamatakis A (2013). "Pruning rogue taxa improves phylogenetic accuracy: an efficient algorithm and webservice." *Systematic Biology*, **62**(1), 162–166. doi: 10.1093/sysbio/sys078. Wilkinson M, Crotti M (2017). "Comments on detecting rogue taxa using RogueNaRok." *Systematics and Biodiversity*, **15**(4), 291–295. doi: 10.1080/14772000.2016.1252440.

See Also

Other tip instability functions: `TipInstability()`

Examples

```
library("TreeTools", quietly = TRUE)
trees <- AddTipEverywhere(BalancedTree(8), 'Rogue')
trees[] <- lapply(trees, AddTip, 'Rogue', 'Rogue2')

sb <- TipVolatility(trees)
sbNorm <- 1 + (99 * (sb - min(sb)) / (max(sb - min(sb))))
col <- hcl.colors(128, 'inferno')[sbNorm]
plot(consensus(trees), tip.color = col)
plot(ConsensusWithout(trees, names(sb[sb == max(sb)])))
```

Index