Package: Rogue (via r-universe)

October 23, 2026

Title Identify Rogue Taxa in Sets of Phylogenetic Trees

Version 2.2.1.9000

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 tips. 'Rogue' provides an explicitly information-theoretic approach to rogue detection (Smith 2022) <doi:10.1093/sysbio/syab099>, an interface to 'RogueNaRok' (Aberer et al. 2013) <doi:10.1093/sysbio/sys078>, and heuristics for measuring leaf instability.

URL https://ms609.github.io/Rogue/

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), stats, TreeDist (> 2.2.0), TreeTools (>= 1.9.1.9003), utils,

Suggests knitr, PlotTools, rmarkdown, spelling, testthat,

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

Config/Needs/coverage covr

Config/Needs/memcheck devtools

Config/Needs/metadata codemetar

Config/Needs/website pkgdown

Config/roxygen2/version 8.0.0

RdMacros Rdpack

SystemRequirements C++17

ByteCompile true

Encoding UTF-8

Language en-GB

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

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

Date/Publication 2026-10-05 06:26:03 UTC

RemoteUrl https://github.com/ms609/Rogue

RemoteRef HEAD

RemoteSha 98f1600f7f4a23e62a1f720181ea62462a7379ea

Contents

RogueTaxa 2
TipInstability 7
TipVolatility 9

Index **11**

`RogueTaxa` *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

```
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",
  p = 0.5,
  log = TRUE,
  average = "median",
  deviation = "mad",
  neverDrop,
  fullSeq = FALSE,
  parallel = FALSE,
  .prepared = FALSE
)

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

Arguments

`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.
`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.
`neverDrop`	Tip labels that should not be dropped from the consensus.
`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.
`verbose`	Logical specifying whether to display output from RogueNaRok. If `FALSE`, output will be included as an attribute of the return value.
`p`	Proportion of trees that must contain a split before it is included in the consensus under consideration. 0.5, the default, corresponds to a majority rule tree; 1.0 will maximize the information content of the strict consensus.
`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`).
`parallel`	Logical, retained for backwards compatibility; the fused C implementation is single-threaded, so this argument is now ignored.
`.prepared`	Logical; if `TRUE`, skip internal tree preparation. For internal use.
`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.
`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.

Details

"Rogue" or (loosely) "wildcard" taxa (Nixon and Wheeler 1992) are leaves whose position in a tree is poorly constrained, typically because much of the phylogenetic data associated with the taxon is either missing or in conflict with other data (Kearney 2002).

These functions use heuristic methods to identify rogue taxa whose removal improves the information content of a consensus tree, by the definitions of information discussed below.

Value

`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`.

`QuickRogue()` returns a `data.frame` in the same format as `RogueTaxa()`.

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

Functions

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

Information criteria

The splitwise phylogenetic information content (SPIC) measure produces the best results (Smith 2022). The SPIC scores a consensus tree by summing the phylogenetic information content of each split, where a split's information content quantifies how much knowing that it is true (with probability estimated from its frequency in the tree set) reduces the entropy over the set of possible tree topologies. The same holds for the mutual clustering information measure; this measure is less obviously suited to the detection of rogues. This measure interprets split frequency as a proxy for the probability that a split is true, which is a valid interpretation of a Bayesian posterior sample (Holder et al. 2008), a reasonable but imperfect interpretation of a bootstrap sample (Berry and Gascuel 1996), and a bad interpretation of a sample of most parsimonious trees.

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.

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.

Berry V, Gascuel O (1996). "On the interpretation of bootstrap trees: appropriate threshold of clade selection and induced gain." *Molecular Biology and Evolution*, **13**(7), 999–1011. doi:10.1093/molbev/13.7.999.

Holder MT, Sukumaran J, Lewis PO (2008). "A justification for reporting the majority-rule consensus tree in Bayesian phylogenetics." *Systematic Biology*, **57**(5), 814–821. doi:10.1080/10635150802422308.

Kearney M (2002). "Fragmentary taxa, missing data, and ambiguity: mistaken assumptions and

conclusions." *Systematic Biology*, **51**(2), 369–381. doi:10.1080/10635150252899824.

Nixon KC, Wheeler QD (1992). "Extinction and the origin of species." In Novacek MJ, Wheeler QD (eds.), *Extinction and phylogeny*, chapter Extinction and the origin of species, 119-143. Columbia University Press, New York.

Smith MR (2022). "Using information theory to detect rogue taxa and improve consensus trees." *Systematic Biology*, **71**(5), 986–1008. doi:10.1093/sysbio/syab099.

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

```
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)

bootTrees <- system.file("example/150.bs", package = "Rogue")
tmpDir <- tempdir()
XX <- capture.output( # Don't print verbose run details to console
  C_RogueNaRok(bootTrees, workDir = tmpDir)
)

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

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

setwd(oldwd)
```

`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,
  parallel = FALSE
)

ColByStability(
  trees,
  log = TRUE,
  average = "mean",
  deviation = "sd",
  pal = hcl.colors(131, "inferno")[1:101]
)
```

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.
`parallel`	Logical, retained for backwards compatibility; the fused C implementation is single-threaded, so this argument is now ignored.
`pal`	A vector listing a sequence of colours to be used for plotting. The earliest entries will be assigned to the most stable tips.

Details

Smith (2022) defines the *instability* of a pair of leaves as the median absolute divergence in the graph geodesic (the number of edges in the shortest path between the leaves) across all trees, normalized

against the mean graph geodesic. 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.

Other concepts of leaf instability include

- The "taxonomic instability index", as implemented in Mesquite: described by Thomson and Shaffer (2010) as $\sum_{(x,y),j \neq i} \frac{|D\ ijx\ - D\ ijy\ |}{(D\ ijx\ - D\ ijy\ )^2}$, where $D\ ijx$ is the patristic distance (i.e. length of edges) between leaves i and j in tree x.
- the average stability of triplets (i.e. quartets including the root) that include the leaf (Thorley and Wilkinson 1999), implemented in "Phyutility" (Smith and Dunn 2008); and related to "positional congruence" measures (Estabrook 1992; Pol and Escapa 2009).

Value

`ColByStability()` returns a named character vector that assigns a colour to each leaf in `trees` according to their stability.

Author(s)

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

References

Estabrook GF (1992). "Evaluating undirected positional congruence of individual taxa between two estimates of the phylogenetic tree for a group of taxa." *Systematic Biology*, **41**(2), 172–177. doi:10.1093/sysbio/41.2.172.

Pol D, Escapa IH (2009). "Unstable taxa in cladistic analysis: identification and the assessment of relevant characters." *Cladistics*, **25**(5), 515–527. doi:10.1111/j.10960031.2009.00258.x.

Smith MR (2022). "Using information theory to detect rogue taxa and improve consensus trees." *Systematic Biology*, **71**(5), 986–1008. doi:10.1093/sysbio/syab099.

Smith SA, Dunn CW (2008). "Phyutility: a phyloinformatics tool for trees, alignments and molecular data." *Bioinformatics*, **24**(5), 715–716. doi:10.1093/bioinformatics/btm619.

Thomson RC, Shaffer HB (2010). "Sparse supermatrices for phylogenetic inference: taxonomy, alignment, rogue taxa, and the phylogeny of living turtles." *Systematic Biology*, **59**(1), 42–58. doi:10.1093/sysbio/syp075.

Thorley JL, Wilkinson M (1999). "Testing the phylogenetic stability of early tetrapods." *Journal of Theoretical Biology*, **200**(3), 343–344. doi:10.1006/jtbi.1999.0999.

See Also

Other tip instability functions: `TipVolatility()`

Examples

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

# Generate some trees with a rogue taxon
trees <- AddTipEverywhere(BalancedTree(8), "Rogue")[3:6]

# Display the strict consensus
plot(consensus(trees), tip.col = ColByStability(trees))

# Add a legend for the colour scale used
PlotTools::SpectrumLegend(
  "bottomleft", bty = "n", # No box
  legend = c("Unstable", "", "Stable"),
  palette = hcl.colors(131, "inferno")[1:101]
)

# Calculate leaf stability
instab <- TipInstability(trees, log = FALSE, ave = "mean", dev = "mad")

# Plot a consensus that omits the least stable leaves
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 (Smith 2020) between trees when that tip is removed. Effective when the number of trees is small.

Usage

```
TipVolatility(trees)
```

Arguments

`trees`	List of trees to analyse.

Value

`TipVolatility()` returns a named vector listing the volatility index calculated for each leaf. Higher values indicate more volatile leaves.

References

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

See Also

Other tip instability functions: TipInstability()

Examples

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

# Generate some trees with two rogue taxa
trees <- AddTipEverywhere(BalancedTree(8), "Rogue")
trees[] <- lapply(trees, AddTip, "Rogue", "Rogue2")

# Calculate tip volatility
sb <- TipVolatility(trees)

# Use volatility to colour leaves in consensus tree
sbNorm <- 1 + (99 * (sb - min(sb)) / (max(sb - min(sb))))
col <- hcl.colors(128, "inferno")[sbNorm]
plot(consensus(trees), tip.color = col)

# Add a legend for the colour scale used
PlotTools::SpectrumLegend(
  "bottomleft", bty = "n", # Suppress box
  inset = -0.02,           # Avoid overlap
  title = "Volatility",
  legend = signif(seq(max(sb), min(sb), length.out = 4), 3),
  palette = hcl.colors(128, "inferno")
)

# Plot consensus after removing highly volatile taxa
plot(ConsensusWithout(trees, names(sb[sb == max(sb)])))
```

Index

∗ **tip instability functions**
    TipInstability, 7
    TipVolatility, 9

C_RogueNaRok *(*RogueTaxa*)*, 2
ColByStability *(*TipInstability*)*, 7

QuickRogue *(*RogueTaxa*)*, 2

RogueTaxa, 2

TipInstability, 7
TipInstability(), *10*
TipVolatility, 9
TipVolatility(), *8*