Package 'phyloscannerR'

April 4, 2018

Title Phylogenetics between and within hosts at once, all along the genome

Version 1.5.2

Description An R package for the second half of phyloscanner (tree analysis).

Depends R (>= 3.4.0)

Imports ape, argparse, data.table (>= 1.10.4-3), dplyr, dtplyr, extraDistr, ff, GGally, ggplot2, ggtree, grid, gridExtra, gtable, kimisc, network, pegas, phangorn, phytools, prodlim, RColorBrewer, reshape2, scales, sna

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

R topics documented:

`draw.summary.statistics` *Graph summary statistics for a single host*

Description

Graph summary statistics for a single host

Usage

```
draw.summary.statistics(phyloscanner.trees, sum.stats, host, verbose = F)
```

Arguments

phyloscanner.trees	A list of class phyloscanner.trees
sum.stats	The output of a call to gather.summary.statistics.
host	The host to obtain graphs for.
verbose	Verbose output

---

gather.summary.statistics

Make a data.table *of per-window host statistics*

Description

This function collects per-window statistics on hosts

Usage

```
gather.summary.statistics(phyloscanner.trees,
  hosts = all.hosts.from.trees(phyloscanner.trees),
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$", verbose = F)
```

Arguments

phyloscanner.trees	A list of class phyloscanner.trees
hosts	A list of hosts to record statistics for. If not specified, every identifiable host in phyloscanner.trees
tip.regex	Regular expression identifying tips from the dataset. This expects up to three capture groups, for host ID, read ID, and read count (in that order). If the latter two groups are missing then read information will not be used. The default matches input from the phyloscanner pipeline where the host ID is the BAM file name.
verbose	Produce verbose output

Value

A data.table

multinomial.calculations *Calculate parameters of the posterior density for pairwise host relationships*

Usage

```
multinomial.calculations(phyloscanner.trees, close.threshold, prior.keff = 3,
  prior.neff = 4, prior.calibrated.prob = 0.66,
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$", allow.mt = F,
  min.reads = 0, min.tips = 0, distant.threshold = close.threshold,
  relationship.types = c("TYPE_PAIR_DI2", "TYPE_PAIR_TO", "TYPE_PAIR_TODI2x2",
  "TYPE_PAIR_TODI2", "TYPE_DIR_TODI2", "TYPE_NETWORK_SCORES",
  "TYPE_ADJ_NETWORK_SCORES", "TYPE_CHAIN_TODI"), verbose = F)
```

Arguments

phyloscanner.trees	A list of class phyloscanner.trees produced by phyloscanner.analyse.trees.
close.threshold	The (potentially normalised) patristic threshold used to determine if two patients' subgraphs are "close"
tip.regex	The regular expression used to identify host IDs in tip names
allow.mt	If FALSE, directionality is only inferred between pairs of hosts where a single clade from one host is nested in one from the other; this is more conservative.
min.reads	The minimum number of reads from a host in a window needed in order for that window to count in determining relationships involving that patient
min.tips	The minimum number of tips from a host in a window needed in order for that window to count in determining relationships involving that patient
distant.threshold	If present, a second distance threshold determines hosts that are "distant" from each other, with those lying between close.threshold and dist.threshold classed as "intermediate". The default is the same as close.threshold, so the intermediate class does not exist.
verbose	Verbose output

Value

A list with two items: dwin giving information on the genome windows for each pair of hosts, and rplkl giving information on phylogenetic relationships between each pair of hosts.

`multipage.summary.statistics`
Draw summary statistics to file for many hosts as a multipage file

---

Description

Draw summary statistics to file for many hosts as a multipage file

Usage

```
multipage.summary.statistics(phyloscanner.trees, sum.stats,
  hosts = all.hosts.from.trees(phyloscanner.trees), file.name,
  height = 11.6929, width = 8.26772, verbose = F)
```

Arguments

`phyloscanner.trees`	A list of class `phyloscanner.trees`
`sum.stats`	The output of a call to `gather.summary.statistics`.
`hosts`	A vector of hosts to obtain graphs for. By default, all hosts detected in `phyloscanner.trees`.
`file.name`	Output file name (should have a .pdf file extension)
`height`	The height of each page of the output file in inches (defaults to A4 size)
`width`	The width of each page of the output file in inches (defaults to A4 size)
`verbose`	Verbose output

`phyloscanner.analyse.trees`
Perform a phyloscanner analysis on a tree or set of trees

Description

These functions perform a parsimony reconstruction and classification of pairwise host relationships.

Usage

```
phyloscanner.analyse.trees(tree.file.directory,
  tree.file.regex = "^RAxML_bestTree.InWindow_([0-9]+_to_[0-9]+)\\.tree$",
  splits.rule = c("s", "r", "f"), sankoff.k = 0,
  sankoff.unassigned.switch.threshold = 0,
  continuation.unassigned.proximity.cost = 1000, outgroup.name = NULL,
  multifurcation.threshold = -1, guess.multifurcation.threshold = F,
  user.blacklist.directory = NULL, user.blacklist.file.regex = NULL,
  duplicate.file.directory = NULL,
 duplicate.file.regex = "^DuplicateReadCountsProcessed_InWindow_([0-9]+_to_[0-9]+).csv$",
  recombination.file.directory = NULL,
 recombination.file.regex = "^RecombinantReads_InWindow_([0-9]+_to_[0-9]+).csv$",
```

```
  alignment.file.directory = NULL, alignment.file.regex = NULL,
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^\\D*([0-9]+)_to_([0-9]+)\\D*$",
  seed = sample(1:1e+07, 1), norm.ref.file.name = NULL,
  norm.standardise.gag.pol = F, norm.constants = NULL,
  parsimony.blacklist.k = 0, raw.blacklist.threshold = 0,
  ratio.blacklist.threshold = 0, do.dual.blacklisting = F,
  max.reads.per.host = Inf, blacklist.underrepresented = F, use.ff = F,
  prune.blacklist = F, count.reads.in.parsimony = T, verbosity = 0,
  no.progress.bars = F)

phyloscanner.analyse.tree(tree.file.name, splits.rule = c("s", "r", "f"),
  sankoff.k = 0, sankoff.unassigned.switch.threshold = 0,
  continuation.unassigned.proximity.cost = 1000, outgroup.name = NULL,
  multifurcation.threshold = -1, guess.multifurcation.threshold = F,
  user.blacklist.file.name = NULL, duplicate.file.name = NULL,
  recombination.file.name = NULL, alignment.file.name = NULL,
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^\\D*([0-9]+)_to_([0-9]+)\\D*$",
  seed = sample(1:1e+07, 1), norm.ref.file.name = NULL,
  norm.standardise.gag.pol = F, norm.constants = NULL,
  parsimony.blacklist.k = 0, raw.blacklist.threshold = 0,
  ratio.blacklist.threshold = 0, do.dual.blacklisting = F,
  max.reads.per.host = Inf, blacklist.underrepresented = F, use.ff = F,
  prune.blacklist = F, count.reads.in.parsimony = T, verbosity = 0,
  no.progress.bars = F)

phyloscanner.generate.blacklist(tree.file.directory,
  tree.file.regex = "^RAxML_bestTree.InWindow_([0-9]+_to_[0-9]+)\\.tree$",
  outgroup.name = NULL, multifurcation.threshold = -1,
  guess.multifurcation.threshold = F, user.blacklist.directory = NULL,
  user.blacklist.file.regex = NULL, duplicate.file.directory = NULL,
 duplicate.file.regex = "^DuplicateReadCountsProcessed_InWindow_([0-9]+_to_[0-9]+).csv$",
  alignment.file.directory = NULL, alignment.file.regex = NULL,
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^\\D*([0-9]+)_to_([0-9]+)\\D*$",
  seed = sample(1:1e+07, 1), norm.ref.file.name = NULL,
  norm.standardise.gag.pol = F, norm.constants = NULL,
  parsimony.blacklist.k = 0, raw.blacklist.threshold = 0,
  ratio.blacklist.threshold = 0, do.dual.blacklisting = F,
  max.reads.per.host = Inf, blacklist.underrepresented = F,
  count.reads.in.parsimony = F, verbosity = 0)
```

Arguments

`tree.file.directory`
: The directory containing all input trees.

`tree.file.regex`
: A regular expression identifying every file in `tree.file.directory` that is to be included in the analysis. The first capture group, if present, gives a unique string identifying each tree. If this is NULL then phyloscanner will attempt to open every file in `tree.file.directory`.

`splits.rule` The rules by which the sets of hosts are split into groups in order to ensure that all groups can be members of connected subgraphs without causing conflicts. Options: s=Sankoff with optional within-host diversity penalty (slow, rigorous, recommended), r=Romero-Severson (quick, less rigorous with >2 hosts), f=Sankoff with continuation costs (experimental).

`sankoff.k` For `splits.rule = s` or `f` only. The k parameter in the Sankoff reconstruction, representing the within-host diversity penalty.

`sankoff.unassigned.switch.threshold`
For `splits.rule = s` only. Threshold at which a lineage reconstructed as infecting a host will transition to the unassigned state, if it would be equally parsimonious to remain in that host.

`continuation.unassigned.proximity.cost`
For `splits.rule = f` only. The branch length at which an node is reconstructed as unassigned if all its neighbouring nodes are a greater distance away. The default is 1000, intended to be effectively infinite, such a node will never normally receive the unassigned state.

`outgroup.name` The name of the tip in the phylogeny/phylogenies to be used as outgroup (if unspecified, trees will be assumed to be already rooted). This should be sufficiently distant to any sequence obtained from a host that it can be assumed that the MRCA of the entire tree was not a lineage present in any sampled individual.

`multifurcation.threshold`
If specified, branches shorter than this in the input tree will be collapsed to form multifurcating internal nodes. This is recommended; many phylogenetics packages output binary trees with short or zero-length branches indicating multifurcations.

`guess.multifurcation.threshold`
Whether to guess the multifurcation threshold from the branch lengths of the trees and the width of the genomic window (if that information is available). It is recommended that trees are examined by eye to check that they do appear to have multifurcations if using this option.

`user.blacklist.directory`
An optional path for a folder containing pre-existing blacklist files. These tips are specified by the user to be excluded from the analysis.

`user.blacklist.file.regex`
A regular expression identifying every file in `user.blacklist.directory` that contains a blacklist. If a capture group is specified then its contents will uniquely identify the tree it belongs to, which must matches the IDs found by `tree.file.regex`. If these IDs cannot be identified then matching will be attempted using genome window coordinates.

`duplicate.file.directory`
An optional path for a folder containing information on duplicate reads, to be used for duplicate blacklisting. Normally this is produced by `phyloscanner_make_trees.py`.

`duplicate.file.regex`
A regular expression identifying every file in `duplicate.file.directory` that contains a duplicates file. If a capture group is specified then its contents will uniquely identify the tree it belongs to, which must matches the IDs found by `tree.file.regex`. If these IDs cannot be identified then matching will be attempted using genome window coordinates.

`recombination.file.directory`
An optional path for a folder containing results of the `phyloscanner_make_trees.py` recombination metric analysis.

`recombination.file.regex`
: A regular expression identifying every file in `recombination.file.directory` that contains a recombination file. If a capture group is specified then its contents will uniquely identify the tree it belongs to, which must matches the IDs found by `tree.file.regex`. If these IDs cannot be identified then matching will be attempted using genome window coordinates.

`alignment.file.regex`
: A regular expression identifying every file in `alignment.directory` that is an alignment. If a capture group is specified then its contents will uniquely identify the tree it belongs to, which must matches the IDs found by `tree.file.regex`. If these IDs cannot be identified then matching will be attempted using genome window coordinates.

`tip.regex`
: Regular expression identifying tips from the dataset. This expects up to three capture groups, for host ID, read ID, and read count (in that order). If the latter two groups are missing then read information will not be used. The default matches input from the phyloscanner pipeline where the host ID is the BAM file name.

`file.name.regex`
: Regular expression identifying window coordinates. Two capture groups: start and end; if the latter is missing then the first group is a single numerical identifier for the window. The default matches input from the phyloscanner pipeline.

`seed`
: Random number seed; used by the downsampling process, and also ties in some parsimony reconstructions can be broken randomly.

`norm.ref.file.name`
: Name of a file giving a normalisation constant for every genome position. Cannot be used simultaneously with `norm.constants`. If neither is given then no normalisation will be performed.

`norm.standardise.gag.pol`
: Use only if `norm.ref.file.name` is given. An HIV-specific option: if true, the normalising constants are standardised so that the average on gag+pol equals 1. Otherwise they are standardised so the average on the whole genome equals 1.

`norm.constants`
: Either the path of a CSV file listing the file name for each tree (column 1) and the respective normalisation constant (column 2) or a single numerical normalisation constant to be applied to every tree. Cannot be used simultaneously with `norm.ref.file.name`. If neither is given then no normalisation will be performed.

`parsimony.blacklist.k`
: The k parameter of the single-host Sankhoff parsimony reconstruction used to identify probable contaminants. A value of 0 is equivalent to not performing parsimony blacklisting.

`raw.blacklist.threshold`
: Used to specify a read count to be used as a raw threshold for duplicate or parsimony blacklisting. Use with `parsimony.blacklist.k` or `duplicate.file.regex` or both. Parsimony blacklisting will blacklist any subgraph with a read count strictly less than this threshold. Duplicate blacklisting will black list any duplicate read with a count strictly less than this threshold. The default value of 0 means nothing is blacklisted.

`ratio.blacklist.threshold`
: Used to specify a read count ratio (between 0 and 1) to be used as a threshold for duplicate or parsimony blacklisting. Use with `parsimony.blacklist.k` or

duplicate.file.regex or both. Parsimony blacklisting will blacklist a subgraph if the ratio of its read count to the total read count from the same host is strictly less than this threshold. Duplcate blacklisting will blacklist a duplicate read if the ratio of its count to the count of the duplicate (from another host) is strictly less than this threshold.

do.dual.blacklisting
: Blacklist all reads from the minor subgraphs for all hosts established as dual by parsimony blacklisting (which must have been done for this to do anything).

max.reads.per.host
: Used to turn on downsampling. If given, reads will be blacklisted such that read counts (or tip counts if no read counts are identified) from each host are equal (although see blacklist.underrepresented.

blacklist.underrepresented
: If TRUE and max.reads.per.host is given, blacklist hosts from trees where their total tip count does not reach the maximum.

use.ff
: Use the ff package to store parsimony reconstruction matrices. Use if you run out of memory.

prune.blacklist
: If TRUE, all blacklisted and reference tips (except the outgroup) are pruned away before starting parsimony-based reconstruction.

count.reads.in.parsimony
: If TRUE, read counts on tips will be taken into account in parsimony reconstructions at the parents of zero-length terminal branches. Not applicable for the Romero-Severson-like reconstruction method.

verbosity
: The type of verbose output. 0=none, 1=minimal, 2=complete

no.progress.bars
: Hide the progress bars from verbose output.

tree.file.name
: The name of a single tree file (Newick or NEXUS format).

user.blacklist.file.name
: The path of a single text file containing the user-specified list of tips to be blacklisted

duplicate.file.name
: The path of a single .csv file specifying which tree tips are from duplicate reads. Normally this is produced by phyloscanner_make_trees.py.

recombination.file.name
: The path for a single file containing the results of the phyloscanner_make_trees.py recombination metric analysis.

alignment.directory
: The directory containing the alignments used to construct the phylogenies.

Details

phyloscanner.analyse.tree is for a single phylogeny and phyloscanner.analyse.trees for a collection, while phyloscanner.generate.blacklist performs the blacklisting steps only.

Value

A list of class phyloscanner.trees. Each element of this list is itself a list of class phyloscanner.tree and corresponds to a single tree, recording details of the phyloscanner reconstruction. The names of the phyloscanner.trees object are the tree IDs, usually derived from file suffixes. A list of class phyloscanner.tree may, depending on exact circumstances, have the following items:

- id The tree ID.
- tree The tree as a phylo object. This will have been rooted and have multifurcations collapsed as requested, but branch lengths are original. It may have been pruned of blacklisted tips if prune.blacklist was specified.
- alignment The alignment as a DNAbin object.
- tree.file.name The file name from which the tree was loaded.
- alignment.file.name The file name for the alignment.
- user.blacklist.file.name The file name for the user-specified blacklist.
- duplicate.file.name The file name for the list of between-host duplicate tips.
- recombination.file.name The file name for the results of the phyloscanner_make_trees.py recombination metric analysis.
- index The index of this tree in the phyloscanner.trees list.
- bl.report A data.frame outlining the blacklisted tips in this tree and the reasons they were blacklisted.
- window.coords A vector giving the start and end of the genome cooardinates of the window from which the tree was built (if the windowed approach was used).
- xcoord A single genome position to locate this tree along the genome; generally the window midpoint in the windowed approach.
- duplicate.file.name The file name used to determine between-host duplicate tips
- original.tip.labels Blacklisting may lead to the pruinig of tips from the tree or their renaming. The original tip labels read from the tree file are recorded here.
- hosts.for.tips A vector mapping each tip onto its correspoinding hosts. Blacklisted tips are given NA.
- normalisation.constant The normalisation constant for this tree. This will be 1 if no normalisation was requested.
- duplicate.tips A list whose entries are vectors of tips whose sequences are exactly alike.
- blacklist A vector of numbers for all tips blacklisted for whatever reason. If the blacklist was pruned away, this will be empty.
- dual.detection.splits A data.frame determining the multiplicity of infection for each host as determined by parsimony blacklisting.
- duals.info A data.frame describing the subgraphs that each tip belong to in the dual infection detection, prior to parsimony and dual blacklisting.
- tips.for.hosts A list giving the tips numbers corresponding to each host
- read.counts A vector giving the read counts for each tip. Blacklisted tips and the outgroup have NAs. All non-NAs will be 1 if the data has no read count.
- splits.table A data frame giving the host and subgraph containing each tip, according to the parsimony reconstruction.
- clades.by.host A list of lists of tips, each determining a monophyletic clade from one host.
- clade.mrcas.by.host A list of vectors containing the MRCA nodes of those clades.
- classification.results A data.frame desribing the pairwise topological classification of each pair of hosts in the tree.

A phyloscanner.trees object has the following attributes:

- readable.coords TRUE if genome window coordinates could be obtained from file names.

- match.mode Either "ID" (tree IDs were identified using tree.file.regex), "coords" (tree IDs were identified from what appear to be genome window coordinates in file names) or "none" (string IDs could not be determined).
- has.read.counts TRUE if phyloscanner detected read counts in tip labels.
- outgroup.name The tip label of the outgroup.

`reconstruct.ancestral.sequences`
Reconstruct the ancestral sequence at every node of the tree

Description

Reconstruct the ancestral sequence at every node of the tree

Usage

```
reconstruct.ancestral.sequences(phyloscanner.tree, verbose = F, default = F,
  ...)
```

Arguments

phyloscanner.tree	A list of class phyloscanner.tree (usually an item in a list of class phyloscanner.trees)
verbose	Verbose output
default	If TRUE, the reconstruction is done according to the default model used in RAxML to build trees for phyloscanner. The ... below will be ignored.
...	Further arguments to be passed to pml and optim.pml

Value

An alignment of the sequences at all nodes (in DNAbin format)

`reconstruct.host.ancestral.sequences`
Find the ancestral sequence at the MRCA of the tips from this host, or, if a dual infection was previously identified, of the MRCA of the tips making up each infection event

---

Description

Find the ancestral sequence at the MRCA of the tips from this host, or, if a dual infection was previously identified, of the MRCA of the tips making up each infection event

Usage

```
reconstruct.host.ancestral.sequences(phyloscanner.tree, host,
  individual.duals = F, verbose = F)
```

Arguments

phyloscanner.tree	A list of class phyloscanner.tree (usually an item in a list of class phyloscanner.trees). This must have an ancestral.alignment element (see *reconstruct.ancestral.sequences*)
host	The host ID
individual.duals	Whether to output multiple sequences for host based on the results of a previous dual infection analysis
verbose	Verbose output

---

simplified.transmission.summary *Simplfy and visually display the pairwise host relationships across all trees*

Description

Simplfy and visually display the pairwise host relationships across all trees

Usage

```
simplified.transmission.summary(phyloscanner.trees, transmission.summary,
  arrow.threshold, plot = F)
```

Arguments

phyloscanner.trees	A list of class phyloscanner.trees
arrow.threshold	The proportion of trees in which a pair of hosts need to show a direction of transmission for that direction to be indicated as an arrow. If both directions meet this threshold, the arrow is in the direction with the larger proportion of trees.
plot	If TRUE, the returned list has an item called simp.diagram, a ggplot object plotting the simplified relationship diagram.
trans.summary	The output of transmission.summary; a data.table.

---

transmission.summary *Summarise the pairwise host relationships across all trees*

Description

Summarise the pairwise host relationships across all trees

Usage

```
transmission.summary(phyloscanner.trees, win.threshold = 0,
  dist.threshold = Inf, allow.mt = T, close.sib.only = F, verbose = F)
```

Arguments

phyloscanner.trees	A list of class phyloscanner.trees
win.threshold	The proportion of windows that a pair of hosts need to be related (adjacent and within dist.threshold of each other) in order for them to appear in the summary.
dist.threshold	The patristic distance within which the subgraphs from two hosts need to be in order for them to be declared related (default is infinity, so adjacent hosts are always related).
allow.mt	If FALSE, directionality is only inferred between pairs of hosts where a single clade from one host is nested in one from the other; this is more conservative.
close.sib.only	If TRUE, then the distance threshold applies only to hosts on sibiling clades. Any ancestry is automatically a relationship.
verbose	Give verbose output

Value

A data.table, every line of which counts the number of pairwise relationships of a particular type between a pair of hosts

---

write.annotated.tree *Write the phylogeny with reconstructed host annotations to file*

Description

Write the phylogeny with reconstructed host annotations to file

Usage

```
write.annotated.tree(phyloscanner.tree, file.name, format = c("pdf", "nex"),
  pdf.scale.bar.width = 0.01, pdf.w = 50, pdf.hm = 0.15, verbose = F)
```

Arguments

phyloscanner.tree	A list of class phyloscanner.tree (usually an item in a list of class phyloscanner.trees)
file.name	The name of the output file
format	The format - PDF or NEXUS - in which to write the output.
pdf.scale.bar.width	The width, in substitutions per site, of the scale bar in PDF output
pdf.w	The width of the output PDF file, in inches
pdf.hm	The height, in inches per tip, of the output PDF file
verbose	Verbose output

Index