Package 'phyloscannerR'

March 3, 2020

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

Version 1.8.0

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

Depends R (>= 3.4.0)

Imports ape, argparse, extraDistr, ff, GGally, ggtree, grid, gtable, kimisc, network, pegas, phangorn, phytools, prodlim, RColorBrewer, reshape2, scales, sna, tidyverse, treeio (>= 1.6.2), viridis

License GPL

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

R topics documented:

assign.groups.for.batched.phyloscanner.analysis

Group individuals for batched phyloscanner analysis

Description

This function groups individuals for phyloscanner analyses, so that phylogenetic linkage between every pair of individuals is assessed at least once. Specifically, individuals are grouped into batches of specified size, and then, all possible pairs of batches are formed. Each of these pairs of batches defines a group of individuals between whom phylogenetic linkages are assessed in one phyloscanner run. The number of individuals in each group is twice the batch size.

Usage

```
assign.groups.for.batched.phyloscanner.analysis(x, batch.size = 50)
```

Arguments

x	Character vector of individual identifiers.
batch.size	Batch size. Default is 50.

Value

tibble with rows 'IND' (individual identifiers), 'PTY_RUN' group for phyloscanner analysis, and 'BATCH' batch of individuals (not used further, but there should be two batches of individuals in each phyloscanner analysis).

Author(s)

Oliver Ratmann

Examples

```
x <- c("15-01402","15-04719","16-00616","16-00801","16-01173","16-01191","16-01302","16-01408","16-01414","
pty.runs <- phyloscannerR::assign.groups.for.batched.phyloscanner.analysis(x, batch.size=50)
```

classify.pairwise.relationships

Classify pairwise host relationships in deep sequence phylogenies

Description

Classify pairwise host relationships in deep sequence phylogenies

Usage

```
classify.pairwise.relationships(
  ptrees,
  close.threshold = 0.025,
  distant.threshold = 0.05,
 relationship.types = c("proximity.3.way", "any.ancestry", "close.x.contiguous",
  "close.and.contiguous", "close.and.adjacent", "close.and.contiguous.and.directed",
    "close.and.adjacent.and.directed", "close.and.contiguous.and.ancestry.cat",
    "close.and.adjacent.and.ancestry.cat"),
  verbose = FALSE
)
```

Arguments

`ptrees`	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".
`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.
`relationship.types`	Classification types. • "proximity.3.way"Classify individuals by phylogenetic distance between subgraphs. Suggested use: to exclude phylogenetic linkage based on distance alone. • "close.and.contiguous"Classify individuals by phylogenetic distance and contiguity of subgraphs. Suggested use: to identify phylogenetically linked pairs. • "close.and.adjacent"Classify individuals by phylogenetic distance and adjacency of subgraphs. Suggested use: to identify phylogenetically linked pairs. • "close.and.contiguous.and.directed"Classify ancestry among contiguous subgraphs. Suggested use: to identify direction of transmission based on contiguous subgraphs. • "close.and.adjacent.and.directed"Classify ancestry among adjacent subgraphs. Suggested use: to identify direction of transmission based on adjacent subgraphs. • "close.and.contiguous.and.ancestry.cat"Classify contiguity and ancestry between individuals. Suggested use: to determine probabilities for transmission networks. • "close.and.adjacent.and.ancestry.cat"Classify adjacency and ancestry between individuals. Suggested use: to determine probabilities for transmission networks.
`verbose`	Verbose output

Value

A data frame with viral phylogenetic classifications of pairwise host relationships in each deep sequence phylogeny

Author(s)

Oliver Ratmann, Matthew Hall

Examples

```
## Not run:
require(phyloscannerR)
#
# Example on data from Rakai Community Cohort Study
# load phyloscanner output from 'phyloscanner.analyse.trees'
#
file <- system.file(file.path('extdata','ptyr192_phsc_analyse_trees_output.RData'),package='phyloscannerR')
load(file) #loads 'phsc', output from 'phyloscanner.analyse.trees'
# use distance thresholds found in analysis of Rakai couples
close.threshold <- 0.025
distant.threshold <- 0.05
# use relationship types based on adjacency
# this also considers linkage etc between individuals who have dual infections, recombinants etc
# ..and thus may not have *all* their subgraphs adjacent to each other
relationship.types <- c('close.and.adjacent',
'close.and.adjacent.and.directed',
'close.and.adjacent.and.ancestry.cat')
dwin <- classify.pairwise.relationships(phsc,
           close.threshold=close.threshold,
           distant.threshold=distant.threshold,
           relationship.types=relationship.types,
           verbose=TRUE)

## End(Not run)
```

---

cmd.phyloscanner.analyse.trees

Make script file for a phyloscanner analysis on a tree or set of trees

---

Description

This function makes a UNIX script file to call `phyloscanner_analyse_trees.R`. Usually, this is useful to parallelise computations; see the Examples.

Usage

```
cmd.phyloscanner.analyse.trees(
  prog.phyloscanner_analyse_trees,
  tree.input,
  control,

   valid.input.args = cmd.phyloscanner.analyse.trees.valid.args(prog.phyloscanner_analyse_trees)
)
```

Arguments

prog.phyloscanner_analyse_trees	The full file name of phyloscanner_analyse_trees.R.
tree.input	One of the following: the name of a single tree file (Newick or NEXUS format); the directory containing all input trees; a zip file containing input trees.
control	List of input arguments to phyloscanner_analyse_trees.
valid.input.args	Vector of valid input arguments.

Value

A character string of UNIX commands.

Author(s)

Oliver Ratmann

See Also

phyloscanner.analyse.trees, cmd.phyloscanner.analyse.trees.valid.args

Examples

```
## Not run:
require(data.table)
require(tidyverse)
require(phyloscannerR)

# specify path to phyloscanner_analyse_trees
prog.phyloscanner_analyse_trees <- '/Users/Oliver/git/phyloscanner/phyloscanner_analyse_trees.R'
# specify out directory
outdir <- '/Users/Oliver/sandbox/DeepSeqProjects/RakaiPopSample_phsc_out190512'
# specify valid input arguments to phyloscanner_analyse_trees
valid.input.args <- cmd.phyloscanner.analyse.trees.valid.args(prog.phyloscanner_analyse_trees)

# set phyloscanner variables
# arguments as used for the Rakai population-based analysis
control <- list()
control$allow.mt <- TRUE
control$alignment.file.directory = NULL
control$alignment.file.regex = NULL
control$blacklist.underrepresented = FALSE
control$count.reads.in.parsimony = TRUE
control$distance.threshold <- '0.025 0.05'
control$do.dual.blacklisting = FALSE
control$duplicate.file.directory = NULL
control$duplicate.file.regex = NULL
control$file.name.regex = "^\\D*([0-9]+)_to_([0-9]+)\\D*$"
control$guess.multifurcation.threshold = FALSE
control$max.reads.per.host = 50
control$min.reads.per.host <- 30
control$min.tips.per.host <- 1
control$multifurcation.threshold = 1e-5
control$multinomial= TRUE
control$norm.constants = NULL
```

```
control$norm.ref.file.name = system.file('HIV_DistanceNormalisationOverGenome.csv',package='phyloscannerR')
control$norm.standardise.gag.pol = TRUE
control$no.progress.bars = TRUE
control$outgroup.name = "REF_CPX_AF460972"
control$output.dir = outdir
control$parsimony.blacklist.k = 20
control$prune.blacklist = FALSE
control$post.hoc.count.blacklisting= TRUE
control$ratio.blacklist.threshold = 0
control$raw.blacklist.threshold = 20
control$recombination.file.directory = NULL
control$recombination.file.regex = NULL
control$relaxed.ancestry = TRUE
control$sankoff.k = 20
control$sankoff.unassigned.switch.threshold = 0
control$seed = 42
control$splits.rule = 's'
control$tip.regex = "^(.*)_fq[0-9]+_read_([0-9]+)_count_([0-9]+)$"
control$tree.file.regex = "^ptyr[0-9]+_InWindow_([0-9]+_to_[0-9]+)\\.tree$"
control$use.ff = FALSE
control$user.blacklist.directory = NULL
control$user.blacklist.file.regex = NULL
control$verbosity = 1

#
# Example 1: make bash for one file
#
tree.input <- system.file(file.path('extdata','Rakai_run192_trees.zip'),package='phyloscannerR')
control$output.string <- 'Rakai_run192'
cmd <- cmd.phyloscanner.analyse.trees(prog.phyloscanner_analyse_trees,
tree.input,
control,
valid.input.args=valid.input.args)
cat(cmd)

#
# Example 2: make bash for many files
#
# download the phyloscanner tree of the Rakai population-based analysis
tmp <- "https://datadryad.org/bitstream/handle/10255/dryad.208473/Dataset_S1.tar?sequence=1"
# specify directory to untar public data
tree.dir <- "RakaiPopSample_deepseqtrees"
# download and untar
download.file(tmp, destfile="Dataset_S1.tar", method="curl")
untar("Dataset_S1.tar", exdir=tree.dir, extras='-xvf')
# list zipped tree files. One zip file contains the viral trees of individuals in one putative transmission netwo
df <- tibble(F=list.files(tree.dir))
df <- df %>%
mutate(TYPE:= gsub('ptyr([0-9]+)_(.*)','\\2', F),
RUN:= as.integer(gsub('ptyr([0-9]+)_(.*)','\\1', F))) %>%
mutate(TYPE:= gsub('^([^\\.]+)\\.[a-z]+$','\\1',TYPE)) %>%
spread(TYPE, F) %>%
set_names(~ str_to_upper(.))
# make one bash script for processing the viral trees of individuals in one putative transmission network.
cmds <- vector('list',nrow(df))
for(i in seq_len(nrow(df)))
{
```

```
control$output.string <- paste0('ptyr',df$RUN[i])
tree.input <- file.path(indir, df$TREES_NEWICK[i])
cmd <- cmd.phyloscanner.analyse.trees(prog.phyloscanner_analyse_trees,
tree.input,
control,
valid.input.args=valid.input.args)
cmds[[i]] <- cmd
}
# output
cat(cmds[[100]])

## End(Not run)
```

`cmd.phyloscanner.analyse.trees.valid.args` *Obtain valid input arguments for a phyloscanner analysis on a tree or set of trees*

Description

Obtain valid input arguments for a phyloscanner analysis on a tree or set of trees

Usage

```
cmd.phyloscanner.analyse.trees.valid.args(prog.phyloscanner_analyse_trees)
```

Arguments

`prog.phyloscanner_analyse_trees`	The full file name of `phyloscanner_analyse_trees.R`.

See Also

`cmd.phyloscanner.analyse.trees`

`count.pairwise.relationships` *Count pairwise relationships across deep-sequence trees*

Description

Count pairwise relationships across deep-sequence trees

Usage

```
count.pairwise.relationships(dwin, w.slide = NA, verbose = TRUE)
```

Arguments

`dwin`	A data frame produced by `classify.pairwise.relationships`.
`w.slide`	Increment between genomic windows. Default: NA.
`verbose`	Verbose output. Default: TRUE.

Value

A data frame with counts of viral phylogenetic classifications between pairs of individuals.

Author(s)

Oliver Ratmann, Matthew Hall

Examples

```
## Not run:
require(phyloscannerR)
#
# continue Rakai example,
# load phyloscanner output from 'phyloscanner.analyse.trees'
#
file <- system.file(file.path('extdata','ptyr192_phsc_analyse_trees_output.R'),package='phyloscannerR')
load(file) #loads 'phsc', output from 'phyloscanner.analyse.trees'
# use distance thresholds found in analysis of Rakai couples
close.threshold <- 0.025
distant.threshold <- 0.05
# use relationship types based on adjacency
# this also considers linkage etc between individuals who have dual infections, recombinants etc
# ..and thus may not have *all* their subgraphs adjacent to each other
relationship.types <- c('proximity.3.way',
'close.and.adjacent',
'close.and.adjacent.and.directed',
'close.and.adjacent.and.ancestry.cat')
dwin <- classify.pairwise.relationships(phsc, close.threshold=close.threshold, distant.threshold=distant.thr
tip.regex <- "^(.*)_fq[0-9]+_read_([0-9]+)_count_([0-9]+)$"
min.reads <- 30
min.tips <- 1
dwin <- select.windows.by.read.and.tip.count(phsc, dwin, tip.regex, min.reads, min.tips)
# count phylogenetic relationships across deep-sequence trees
dc <- count.pairwise.relationships(dwin)
#
#  end of Rakai example
#

## End(Not run)
```

---

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

---

find.bam.and.references

Find bam and corresponding reference files

Description

This function finds bam and corresponding reference files in a given directory, and groups them by a common sample ID as well as by an individual ID.

Usage

```
find.bam.and.references(
  data.dir,
  regex.person = "^([A-Z0-9]+-[A-Z0-9]+)-.*$",
  regex.bam = "^(.*)\\.bam$",
  regex.ref = "^(.*)_ref\\.fasta$",
  verbose = 1
)
```

Arguments

data.dir	Full path of data directory
regex.person	Regular expression with one set of round brackets, which identifies the person ID in the file name of bams and references
regex.bam	Regular expression that identifies bam files, with one set of round brackets that identifies the sample ID.
regex.ref	Regular expression that identifies ref files, with one set of round brackets that identifies the sample ID.

Value

tibble with rows 'IND' (individual identifier), 'SAMPLE' (sample identifier), 'BAM' (bam file), and 'REF' (reference file).

`gather.summary.statistics` *Make a* `tibble` *of per-window host statistics*

Description

This function collects per-window statistics on hosts

Usage

```
## S3 method for class 'summary.statistics'
gather(
  ptrees,
  hosts = all.hosts.from.trees(ptrees),
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  verbose = F
)
```

Arguments

`ptrees`	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 `tibble`

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

Arguments

ptrees	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 ptrees.
file.name	Output file name (expected to be a PDF)
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,
  alignment.format = "fasta",
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^(?:.*\\D)?([0-9]+)_to_([0-9]+).*$",
  seed = sample(1:1e+07, 1),
  norm.ref.file.name = NULL,
  norm.standardise.gag.pol = F,
  norm.constants = NULL,
  allow.mt = F,
  relaxed.ancestry = F,
  parsimony.blacklist.k = 0,
  raw.blacklist.threshold = 0,
  ratio.blacklist.threshold = 0,
  do.dual.blacklisting = F,
  max.reads.per.host = Inf,
  blacklist.underrepresented = F,
  min.reads.per.host = 1,
  min.tips.per.host = 1,
  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,
  alignment.format = "fasta",
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^(?:.*\\D)?([0-9]+)_to_([0-9]+).*$",
  seed = sample(1:1e+07, 1),
  norm.ref.file.name = NULL,
  norm.standardise.gag.pol = F,
  norm.constants = NULL,
  allow.mt = F,
  relaxed.ancestry = F,
  parsimony.blacklist.k = 0,
  raw.blacklist.threshold = 0,
  ratio.blacklist.threshold = 0,
  do.dual.blacklisting = F,
```

```
  max.reads.per.host = Inf,
  blacklist.underrepresented = F,
  min.reads.per.host = 1,
  min.tips.per.host = 1,
  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$",
  alignment.file.directory = NULL,
  alignment.file.regex = NULL,
  alignment.format = "fasta",
  tip.regex = "^(.*)_read_([0-9]+)_count_([0-9]+)$",
  file.name.regex = "^.*([0-9]+)_to_([0-9]+).*$",
  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,
  min.reads.per.host = 1,
  min.tips.per.host = 1,
  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.

`alignment.format`
: The file format for alignment files, as per ape::read.dna. Default is FASTA.

`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.

`allow.mt`
: If FALSE (the default0), 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.

`relaxed.ancestry`
: If TRUE, then an ancestry call requires only that at least one subgraph from one host is descended from the other, and that there are no subgrapghs in the opposite arrangement. If TRUE (the default), then it requires that all subgraphs from one host are descended from one from the other.

`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, tips 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.

min.reads.per.host
: If given, hosts will be entirely blacklisted from a given tree if they have fewer than this number of reads on it (after all other blacklisting except downsampling).

min.tips.per.host
: If given, hosts will be entirely blacklisted from a given tree if they have fewer than this number of tips on it (after all other blacklisting except downsampling).

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.

Examples

```
#
# Example on data from Rakai Community Cohort Study
#
## Not run:

require(phyloscannerR)

# extract RCCS example data
tree.file.zip <- system.file(file.path('extdata','Rakai_run192_trees.zip'),package='phyloscannerR')
tree.file.directory <- tempdir()
unzip(tree.file.zip, exdir=tree.file.directory, junkpaths=TRUE)

# arguments used for RCCS analysis
file.name.regex <- "^\\D*([0-9]+)_to_([0-9]+)\\D*$"
max.reads.per.host <- 50
multifurcation.threshold <- 1e-5
norm.ref.file.name <- system.file('HIV_DistanceNormalisationOverGenome.csv',package='phyloscannerR')
outgroup.name <- "REF_CPX_AF460972"
raw.blacklist.threshold <- 20
sankoff.k <- 20
sankoff.unassigned.switch.threshold <- 0
seed <- 42
splits.rule <- 's'
relaxed.ancestry <- TRUE
allow.mt <- TRUE
tip.regex <- "^(.*)_fq[0-9]+_read_([0-9]+)_count_([0-9]+)$"
tree.file.regex <- "^ptyr192_InWindow_([0-9]+_to_[0-9]+)\\.tree$"
verbosity <- 1

# analyse deep sequence trees
```

```
phsc <- phyloscanner.analyse.trees(tree.file.directory,
             allow.mt=allow.mt,
             alignment.file.directory = NULL,
             alignment.file.regex = NULL,
             blacklist.underrepresented = FALSE,
             count.reads.in.parsimony = TRUE,
             do.dual.blacklisting = FALSE,
             duplicate.file.directory = NULL,
             duplicate.file.regex = NULL,
             file.name.regex = file.name.regex,
             guess.multifurcation.threshold = FALSE,
             max.reads.per.host = max.reads.per.host,
             multifurcation.threshold = multifurcation.threshold,
             norm.constants = NULL,
             norm.ref.file.name = NULL,
             norm.standardise.gag.pol = TRUE,
             no.progress.bars = FALSE,
             outgroup.name = outgroup.name,
             parsimony.blacklist.k = sankoff.k,
             prune.blacklist = FALSE,
             ratio.blacklist.threshold = 0,
             raw.blacklist.threshold = raw.blacklist.threshold,
             recombination.file.directory = NULL,
             recombination.file.regex = NULL,
             relaxed.ancestry = relaxed.ancestry,
             sankoff.k = sankoff.k,
             sankoff.unassigned.switch.threshold = sankoff.unassigned.switch.threshold,
             seed = seed,
             splits.rule = splits.rule,
             tip.regex = tip.regex,
             tree.file.regex = tree.file.regex,
             use.ff = FALSE,
             user.blacklist.directory = NULL,
             user.blacklist.file.regex = NULL,
             verbosity = verbosity
             )

## End(Not run)
```

`produce.pairwise.graphs` *Draw bar graphs of pairwise topological/distance relationships*

Description

Draw bar graphs of pairwise topological/distance relationships

Usage

```
produce.pairwise.graphs(
  ptrees,
  dist.thresh = 0.025,
  hosts = all.hosts.from.trees(ptrees),
  contiguous.pairs = F,
```

```
  inclusion = c("both", "either")
)
```

Arguments

ptrees	A list of class phyloscanner.trees
dist.thresh	The distance threshold used to select likely transmission pairs
hosts	A list of hosts (as a vector) to obtain graphs for. By default, all pairs of hosts detected in ptrees.
contiguous.pairs	If TRUE pairs require contiguous (rather than ajacent) subgraphs to be identified as likely transmissions
inclusion	If "both", then only pairs in which both individuals are members of hosts are included. If "either" then pairs only need have one member from hosts

Value

A list whose elements are data, the underlying data frame for the graph, and graph, the graph itself.

Examples

```
#
# Example on data from Rakai Community Cohort Study
#
## Not run:
file <- system.file(file.path('extdata','ptyr192_phsc_analyse_trees_output.RData'),package='phyloscannerR')
load(file) #loads 'phsc', output from 'phyloscanner.analyse.trees'
hosts <- c('RkA05868F','RkA05968M','RkA00369F','RkA01344M')
inclusion <- "both"
tmp <- produce.pairwise.graphs(phsc, hosts=hosts, inclusion = "both")
tmp$graph

## End(Not run)
```

---

produce.pairwise.graphs2

Draw bar graphs of pairwise topological/distance relationships, version 2

Description

This function generates scan plots that summarize reconstructed viral phylogenetic relationships of two individuals. Several pairs of individuals can be processed simultaneously. For each pair of individuals, the scan plot shows the phylogenetic distance on the y-axis and topological relationship in colours between subgraphs from both individuals in each deep-sequence phylogeny across the genome. The genomic position on the x-axis indicates the start of each read alignment.

Usage

```
produce.pairwise.graphs2(
  ptrees,
  hosts = all.hosts.from.trees(ptrees),
  inclusion = c("both", "either"),
  dwin = NULL,
 control = list(yintercept_close = 0.025, yintercept_dist = 1, breaks_x = seq(0, 10000,
  500), minor_breaks_x = seq(0, 10000, 100), breaks_y = c(0.001, 0.0025, 0.005, 0.01,
  0.025, 0.05, 0.1, 0.25), limits_y = c(0.001, 0.4), fill.topology = c(ancestral =
  "deepskyblue1", descendant = "deepskyblue4", intermingled = "#FDB863", sibling =
   "#8073AC", other = "grey80"))
)
```

Arguments

ptrees	A list of class phyloscanner.trees
hosts	A list of hosts (as a vector) to obtain graphs for. By default, all pairs of hosts detected in ptrees.
inclusion	If "both", then only pairs in which both individuals are members of hosts are included. If "either" then pairs only need have one member from hosts
dwin	Optional output of classify.pairwise.relationships. This can be specified to avoid double calculations.
control	List of plotting attributes.

Value

A list whose elements are data, the underlying data frame for the graph, and graph, the graph itself.

Author(s)

Oliver Ratmann

See Also

classify.pairwise.relationships

Examples

```
#
# Example on data from Rakai Community Cohort Study
# remember that you can specify dwin to save computing time, if you have it already computed
#
## Not run:
file <- system.file(file.path('extdata','ptyr192_phsc_analyse_trees_output.RData'),package='phyloscannerR')
load(file) #loads 'phsc', output from 'phyloscanner.analyse.trees'
hosts <- c('RkA05868F','RkA05968M','RkA00369F','RkA01344M')
inclusion <- "both"
tmp <- produce.pairwise.graphs2(phsc, hosts=hosts, inclusion = "both")
tmp$graph

## End(Not run)
```

`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(ptree, verbose = F, default = F, ...)
```

Arguments

`ptree`	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(
  ptree,
  host,
  individual.duals = F,
  verbose = F
)
```

Arguments

ptree	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

---

select.windows.by.read.and.tip.count

Select for further analysis relationship classifications by read and tip counts

---

Description

Select for further analysis relationship classifications by read and tip counts

Usage

```
## S3 method for class 'windows.by.read.and.tip.count'
select(ptrees, dwin, tip.regex, min.reads, min.tips, verbose = F)
```

Arguments

ptrees	A list of class phyloscanner.trees produced by phyloscanner.analyse.trees.
dwin	A data frame produced by classify.pairwise.relationships.
tip.regex	The regular expression used to identify host IDs in tip names
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
verbose	Verbose output

Value

A data frame with viral phylogenetic classifications of pairwise host relationships in each deep sequence phylogeny

Author(s)

Oliver Ratmann, Matthew Hall

Examples

```
## Not run:
require(phyloscannerR)
#
# continue Rakai example,
# load phyloscanner output from 'phyloscanner.analyse.trees'
#
file <- system.file(file.path('extdata','ptyr192_phsc_analyse_trees_output.R'),package='phyloscannerR')
load(file) #loads 'phsc', output from 'phyloscanner.analyse.trees'
# use distance thresholds found in analysis of Rakai couples
close.threshold <- 0.025
distant.threshold <- 0.05
# use relationship types based on adjacency
# this also considers linkage etc between individuals who have dual infections, recombinants etc
# ..and thus may not have *all* their subgraphs adjacent to each other
relationship.types <- c('proximity.3.way',
'close.and.adjacent',
'close.and.adjacent.and.directed',
'close.and.adjacent.and.ancestry.cat')
dwin <- classify.pairwise.relationships(phsc, allow.mt=TRUE, close.threshold=close.threshold, distant.thresh
tip.regex <- "^(.*)_fq[0-9]+_read_([0-9]+)_count_([0-9]+)$"
min.reads <- 30
min.tips <- 1
dwin <- select.windows.by.read.and.tip.count(phsc, dwin, tip.regex, min.reads, min.tips)
#
#  end of Rakai example
#

## End(Not run)
```

`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(
  ptrees,
  transmission.summary,
  arrow.threshold,
  plot = F
)
```

Arguments

`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.

`phyloscanner.trees` A list of class `phyloscanner.trees`

`trans.summary` The output of `transmission.summary`; a `tibble`.

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

Description

Summarise the pairwise host relationships across all trees

Usage

```
transmission.summary(
  ptrees,
  win.threshold = 0,
  dist.threshold = Inf,
  tip.regex,
  min.tips = 1,
  min.reads = 1,
  close.sib.only = F,
  verbose = F
)
```

Arguments

`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).

`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.

`min.tips` The minimum number of tips that a host must have in each tree for it to be counted in that tree (A legacy option - we recommend using the blacklist functionality.)

`min.reads` The minimum number of reads that a host must have in each tree for it to be counted in that tree (A legacy option - we recommend using the blacklist functionality.)

`close.sib.only` If TRUE, then the distance threshold applies only to hosts in sibling clades. Any ancestry is automatically a relationship.

verbose Give verbose output

phyloscanner.trees
A list of class phyloscanner.trees

Value

A tibble, 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(
  ptree,
  file.name,
  format = c("pdf", "nex"),
  pdf.scale.bar.width = 0.01,
  pdf.w = 50,
  pdf.hm = 0.15,
  verbose = F
)
```

Arguments

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

phyloscanner.tree
A list of class phyloscanner.tree (usually an item in a list of class phyloscanner.trees)

Index