Package 'dowser'

April 13, 2022

Type Package

Version 1.0.0

Date 2022-04-11

Title B Cell Receptor Phylogenetics Toolkit

Description
Provides a set of functions for inferring, visualizing, and analyzing B cell phylogenetic trees.
Provides methods to 1) reconstruct unmutated ancestral sequences,
2) build B cell phylogenetic trees using multiple methods,
3) visualize trees with metadata at the tips,
4) reconstruct intermediate sequences,
5) detect biased ancestor-descendant relationships among metadata types
Workflow examples available at documentation site (see URL).
Citations:
Hoehn et al (2020) <doi:10.1101/2020.05.30.124446>,
Hoehn et al (2021) <doi:10.1101/2021.01.06.425648>.

License AGPL-3

URL https://dowser.readthedocs.io

BugReports https://bitbucket.org/kleinstein/dowser/issues

LazyData true

BuildVignettes true

VignetteBuilder knitr

Encoding UTF-8

biocViews

Depends R (>= 3.1.2), ggplot2 (>= 3.2.0)

Imports alakazam (>= 1.1.0), ape (>= 5.6), Biostrings, dplyr (>= 0.8.1), ggtree, graphics, gridExtra, markdown, methods, phangorn (>= 2.7.1), phylotate, RColorBrewer, rlang, shazam (>= 1.1.0), stats, stringr, tidyselect, tidyr, utils

Suggests knitr, rmarkdown, testthat

RoxygenNote 7.1.2

Collate 'Data.R' 'Dowser.R' 'Clones.R' 'Classes.R' 'Plotting.R' 'Germlines.R' 'Statistics.R' 'TreeFunctions.R'

NeedsCompilation no

Author Kenneth Hoehn [aut, cre], Susanna Marquez [ctb], Jason Vander Heiden [ctb], Steven Kleinstein [aut, cph]

Maintainer Kenneth Hoehn <kenneth.hoehn@yale.edu>

Repository CRAN

Date/Publication 2022-04-13 15:42:29 UTC

R topics documented:

`airrClone-class` *S4 class defining a clone in Dowser*

Description

`airrClone` defines a common data structure for perform lineage recontruction from AIRR data, based heavily on alakazam::ChangeoClone.

Slots

`data` data.frame containing sequences and annotations. Contains the columns `SEQUENCE_ID` and `SEQUENCE`, as well as any additional sequence-specific annotation columns

`clone` string defining the clone identifier

`germline` string containing the heavy chain germline sequence for the clone

`lgermline` string containing the light chain germline sequence for the clone

`hlgermline` string containing the combined germline sequence for the clone

`v_gene` string defining the V segment gene call

`j_gene` string defining the J segment gene call

`junc_len` numeric junction length (nucleotide count)

`locus` index showing which locus represented at each site

`region` index showing FWR/CDR region for each site

`phylo_seq` sequence column used for phylogenetic tree building

`numbers` index (usually IMGT) number of each site in `phylo_seq`

See Also

See formatClones for use.

`bootstrapTrees` *Deprecated! Please use findSwitches instead.*

Description

`bootstrapTrees` Phylogenetic bootstrap function.

Usage

```
bootstrapTrees(
  clones,
  bootstraps,
  nproc = 1,
  trait = NULL,
  dir = NULL,
  id = NULL,
  modelfile = NULL,
  build = "pratchet",
  exec = NULL,
  igphyml = NULL,
  fixtrees = FALSE,
  quiet = 0,
  rm_temp = TRUE,
  palette = NULL,
  resolve = 2,
  rep = NULL,
  keeptrees = TRUE,
  lfile = NULL,
  seq = NULL,
  downsample = FALSE,
  tip_switch = 20,
  boot_part = "locus",
  force_resolve = FALSE,
  ...
)
```

Arguments

`clones`	tibble `airrClone` objects, the output of formatClones
`bootstraps`	number of bootstrap replicates to perform
`nproc`	number of cores to parallelize computations
`trait`	trait to use for parsimony models (required if `igphyml` specified)
`dir`	directory where temporary files will be placed (required if `igphyml` or `dnapars` specified)
`id`	unique identifer for this analysis (required if `igphyml` or `dnapars` specified)

modelfile	file specifying parsimony model to use
build	program to use for tree building (phangorn, dnapars)
exec	location of desired phylogenetic executable
igphyml	location of igphyml executible if trait models desired
fixtrees	keep tree topologies fixed? (bootstrapping will not be perfomed)
quiet	amount of rubbish to print to console
rm_temp	remove temporary files (default=TRUE)
palette	a named vector specifying colors for each state
resolve	how should polytomies be resolved? 0=none, 1=max parsminy, 2=max ambiguity + polytomy skipping, 3=max ambiguity
rep	current bootstrap replicate (experimental)
keeptrees	keep trees estimated from bootstrap replicates? (TRUE)
lfile	lineage file input to igphyml if desired (experimental)
seq	column name containing sequence information
downsample	downsample clones to have a maximum specified tip/switch ratio?
tip_switch	maximum allowed tip/switch ratio if downsample=TRUE
boot_part	is "locus" bootstrap columns for each locus separately
force_resolve	continue even if polytomy resolution fails?
...	additional arguments to be passed to tree building program

Value

A list of trees and/or switch counts for each bootstrap replicate.

---

buildClonalGermline buildClonalGermline *Determine consensus clone sequence and create germline for clone*

Description

Determine consensus clone sequence and create germline for clone

Usage

```
buildClonalGermline(
  receptors,
  references,
  organism = "human",
  locus = "IGH",
  use_regions = FALSE,
  vonly = FALSE,
  seq = "sequence_alignment",
```

```
  id = "sequence_id",
  clone = "clone_id",
  v_call = "v_call",
  j_call = "j_call",
  j_germ_length = "j_germline_length",
  j_germ_aa_length = "j_germline_aa_length",
  amino_acid = FALSE,
  ...
)
```

Arguments

`receptors`	AIRR-table containing sequences from one clone
`references`	Full list of reference segments, see readIMGT
`organism`	Species in `references` being analyzed
`locus`	locus in `references` being analyzed
`use_regions`	Return string of VDJ regions? (optional)
`vonly`	Return germline of only v segment?
`seq`	Column name for sequence alignment
`id`	Column name for sequence ID
`clone`	Column name for clone ID
`v_call`	Column name for V gene segment gene call
`j_call`	Column name for J gene segment gene call
`j_germ_length`	Column name of J segment length within germline
`j_germ_aa_length`	Column name of J segment amino acid length (if amino_acid=TRUE)
`amino_acid`	Perform reconstruction on amino acid sequence (experimental)
`...`	Additional arguments passed to buildGermline

Details

Return object adds/edits following columns:

- `seq`: Sequences potentially padded same length as germline
- `germline_alignment`: Full length germline
- `germline_alignment_d_mask`: Full length, D region masked
- `vonly`: V gene segment of germline if vonly=TRUE
- `regions`: String of VDJ segment in position if use_regions=TRUE

Value

Tibble with reconstructed germlines

See Also

createGermlines buildGermline, stitchVDJ

buildGermline buildGermline *reconstruct germline segments from alignment data*

Description

Reconstruct germlines from alignment data.

Usage

```
buildGermline(
  receptor,
  references,
  seq = "sequence_alignment",
  id = "sequence_id",
  clone = "clone_id",
  v_call = "v_call",
  d_call = "d_call",
  j_call = "j_call",
  v_germ_start = "v_germline_start",
  v_germ_end = "v_germline_end",
  v_germ_length = "v_germline_length",
  d_germ_start = "d_germline_start",
  d_germ_end = "d_germline_end",
  d_germ_length = "d_germline_length",
  j_germ_start = "j_germline_start",
  j_germ_end = "j_germline_end",
  j_germ_length = "j_germline_length",
  np1_length = "np1_length",
  np2_length = "np2_length",
  amino_acid = FALSE
)
```

Arguments

receptor	row from AIRR-table containing sequence of interest
references	list of reference segments. Must be specific to organism and locus
seq	Column name for sequence alignment
id	Column name for sequence ID
clone	Column name for clone ID
v_call	Column name for V gene segment gene call
d_call	Column name for D gene segment gene call
j_call	Column name for J gene segment gene call
v_germ_start	Column name of index of V segment start within germline
v_germ_end	Column name of index of V segment end within germline

v_germ_length	Column name of index of V segment length within germline
d_germ_start	Column name of index of D segment start within germline
d_germ_end	Column name of index of D segment end within germline
d_germ_length	Column name of index of D segment length within germline
j_germ_start	Column name of index of J segment start within germline
j_germ_end	Column name of index of J segment end within germline
j_germ_length	Column name of index of J segment length within germline
np1_length	Column name in receptor specifying np1 segment length
np2_length	Column name in receptor specifying np2 segment length
amino_acid	Perform reconstruction on amino acid sequence (experimental)

Details

Return object contains multiple IMGT-gapped germlines:

- `full`: Full length germline
- `dmask`: Full length germline with D region masked
- `vonly`: V gene segment of germline
- `regions`: String showing VDJ segment of each position

Value

List of reconstructed germlines

See Also

buildClonalGermline, stitchVDJ

`buildIgphyml` *Wrapper to build IgPhyML trees and infer intermediate nodes*

Description

Wrapper to build IgPhyML trees and infer intermediate nodes

Usage

```
buildIgphyml(
  clone,
  igphyml,
  trees = NULL,
  nproc = 1,
  temp_path = NULL,
  id = NULL,
```

```
  rseed = NULL,
  quiet = 0,
  rm_files = TRUE,
  rm_dir = NULL,
  partition = c("single", "cf", "hl", "hlf", "hlc", "hlcf"),
  omega = "e",
  optimize = "lr",
  motifs = "FCH",
  hotness = "e,e,e,e,e,e",
  asrc = 0.95,
  splitfreqs = FALSE,
  ...
)
```

Arguments

clone	airrClone object
igphyml	igphyml executable
trees	list of tree topologies if desired
nproc	number of cores for parallelization
temp_path	path to temporary directory
id	IgPhyML run id
rseed	random number seed if desired
quiet	amount of rubbish to print
rm_files	remove temporary files?
rm_dir	remove temporary directory?
partition	How to partition omegas along sequences (see details)
omega	omega parameters to estimate (see IgPhyML docs)
optimize	optimize HLP rates (r), lengths (l), topology (t)
motifs	motifs to consider (see IgPhyML docs)
hotness	hotness parameters to estimate (see IgPhyML docs)
asrc	Intermediate sequence cutoff probability
splitfreqs	Calculate codon frequencies on each partition separately?
...	Additional arguments (not currently used)

Details

Partition options:

- single: 1 omega for whole sequence
- cf: 2 omegas, 1 for all CDRs and 1 for all FWRs
- hl: 2 omegas, 1 for heavy and 1 for light chain
- hlf: 3 omegas, 1 for all CDRs, 2 for heavy/light FWRs
- hlc: 3 omegas, 1 for all FWRs, 2 for heavy/light CDRs
- hlcf: 4 omegas, 1 for each heavy/light CDR/FWR combination

Value

phylo object created by igphyml with nodes attribute containing reconstructed sequences.

---

buildPhylo *Wrapper for alakazam::buildPhylipLineage*

Description

Wrapper for alakazam::buildPhylipLineage

Usage

```
buildPhylo(
  clone,
  exec,
  temp_path = NULL,
  verbose = 0,
  rm_temp = TRUE,
  seq = "sequence",
  tree = NULL,
  onetree = TRUE
)
```

Arguments

clone	airrClone object
exec	dnapars or dnaml executable
temp_path	path to temporary directory
verbose	amount of rubbish to print
rm_temp	remove temporary files?
seq	sequece column in airrClone object
tree	fixed tree topology if desired (currently does nothing if specified)
onetree	Only sample one tree if multiple found.

Value

phylo object created by dnapars or dnaml with nodes attribute containing reconstructed sequences.

buildPML *Wrapper for phangorn::optim.pml*

Description

Wrapper for phangorn::optim.pml

Usage

```
buildPML(
  clone,
  seq = "sequence",
  sub_model = "GTR",
  gamma = FALSE,
  asr = "seq",
  asr_thresh = 0.05,
  tree = NULL,
  data_type = "DNA",
  optNni = TRUE,
  optQ = TRUE,
  verbose = FALSE
)
```

Arguments

clone	airrClone object
seq	sequece column in airrClone object
sub_model	substitution model to use
gamma	gamma site rate variation?
asr	return sequence or probability matrix?
asr_thresh	threshold for including a nucleotide as an alternative
tree	fixed tree topology if desired.
data_type	Are sequences DNA or AA?
optNni	Optimize tree topology
optQ	Optimize Q matrix
verbose	Print error messages as they happen?

Value

phylo object created by phangorn::optim.pml with nodes attribute containing reconstructed sequences.

`buildPratchet` *Wrapper for phangorn::pratchet*

Description

Wrapper for phangorn::pratchet

Usage

```
buildPratchet(
  clone,
  seq = "sequence",
  asr = "seq",
  asr_thresh = 0.05,
  tree = NULL,
  asr_type = "MPR",
  verbose = 0,
  resolve_random = TRUE,
  data_type = "DNA"
)
```

Arguments

`clone`	`airrClone` object
`seq`	sequece column in `airrClone` object
`asr`	return sequence or probability matrix?
`asr_thresh`	threshold for including a nucleotide as an alternative
`tree`	fixed tree topology if desired.
`asr_type`	MPR or ACCTRAN
`verbose`	amount of rubbish to print
`resolve_random`	randomly resolve polytomies?
`data_type`	Are sequences DNA or AA?

Value

`phylo` object created by phangorn::pratchet with nodes attribute containing reconstructed sequences.

collapseNodes *Collapse internal nodes with the same predicted sequence*

Description

`collapseNodes` Node collapsing function.

Usage

```
collapseNodes(trees, tips = FALSE, check = TRUE)
```

Arguments

`trees`	a tibble of `airrClone` objects, the output of getTrees
`tips`	collapse tips to internal nodes? (experimental)
`check`	check that collapsed nodes are consistent with original tree

Details

Use plotTrees(trees)[[1]] + geom_label(aes(label=node)) + geom_tippoint() to show node labels, and getSeq to return internal node sequences

Value

A tibble with `phylo` objects that have had internal nodes collapsed.

See Also

getTrees

colorTrees *Get a color palette for a predefined set of trait values*

Description

`colorTree` Gets a color palette for a predefined set of trait values

Usage

```
colorTrees(trees, palette, ambig = "blend")
```

Arguments

`trees`	list of phylo objects with assigned internal node states
`palette`	named vector of colors (see getPalette)
`ambig`	how should ambiguous states be colored (blend or grey)

Details

Trees must have node states represented in a "states" vector. By default, ambiguous states (separated by ",") have their colors blended. If

Value

A list of colored trees

See Also

getPalette, getTrees, plotTrees

`condenseTrees` *Condense a set of equally parsimonious node labels into a single tree*

Description

`condenseTrees` Condenses a set of equally parsimonious node labels into a single tree

Usage

```
condenseTrees(trees, states, palette)
```

Arguments

`trees`	List of the same tree with equally parsimonious labels
`states`	States in model
`palette`	Named vector with a color per state

Value

a `phylo` object representing all represented internal node states

correlationTest *Run date randomization test for temporal signal on a set of trees.*

Description

correlationTest performs root-to-tip regression date randomization test

Usage

```
correlationTest(
  clones,
  permutations = 1000,
  minlength = 0.001,
  perm_type = c("clustered", "uniform"),
  time = "time",
  sequence = "sequence_id",
  germline = "Germline",
  verbose = FALSE,
  polyresolve = TRUE,
  alternative = c("greater", "two.sided"),
  storeTree = FALSE,
  nproc = 1
)
```

Arguments

clones	A tibble object containing airrClone and phylo objects
permutations	Number of permutations to run
minlength	Branch lengths to collapse in trees
perm_type	Permute among single timepoint clades or uniformly among tips
time	Column name holding numeric time information
sequence	Column name holding sequence ID
germline	Germline sequence name
verbose	Print lots of rubbish while running?
polyresolve	Resolve polytomies to have a minimum number of single timepoint clades
alternative	Is alternative that the randomized correlation are greater than or equal to observed, or greater/less than?
storeTree	Store the tree used?
nproc	Number of cores to use for calculations. Parallelizes by tree.

Details

Object returned contains these columns which are added or modified from input:

- `data`: airrClone object, same as input but with additional columns "cluster" which correspond to permutation cluster, and "divergence."
- `slope`: Slope of linear regression between divergence and time.
- `correlation`: Correlation between divergence and time.
- `p`: p value of correlation compared to permuted correlations.
- `random_correlation`: Mean correlation of permutation replicates.
- `min_p`: Minimum p value of data, determined by either the number of distinct clade/timepoint combinations or number of permutations.
- `nposs`: Number of possible distinct timepoint/clade combinations.
- `nclust`: Number of clusters used in permutation. If perm_type="uniform" this is the number of tips.
- `p_gt/p_lt`: P value that permuted correlations are greater or less than observed correlation. Only returned if alternative = "two.sided"
- `test_trees`: The phylo tree objects used, possibly with resolved polytomies.

Value

A `tibble` with the same columns as clones, but additional columns corresponding to test statistics for each clone.

See Also

Uses output from `getTrees`.

`createGermlines` *createGermlines Determine consensus clone sequence and create germline for clone*

Description

createGermlines Determine consensus clone sequence and create germline for clone

Usage

```
createGermlines(
  data,
  references,
  organism = "human",
  locus = "IGH",
  nproc = 1,
  seq = "sequence_alignment",
```

```
  v_call = "v_call",
  d_call = "d_call",
  j_call = "j_call",
  amino_acid = FALSE,
  id = "sequence_id",
  clone = "clone_id",
  v_germ_start = "v_germline_start",
  v_germ_end = "v_germline_end",
  v_germ_length = "v_germline_length",
  d_germ_start = "d_germline_start",
  d_germ_end = "d_germline_end",
  d_germ_length = "d_germline_length",
  j_germ_start = "j_germline_start",
  j_germ_end = "j_germline_end",
  j_germ_length = "j_germline_length",
  np1_length = "np1_length",
  np2_length = "np2_length",
  na.rm = TRUE,
  fields = NULL,
  verbose = 0,
  ...
)
```

Arguments

`data`	AIRR-table containing sequences from one clone
`references`	Full list of reference segments, see readIMGT
`organism`	Species in `references` being analyzed
`locus`	locus in `references` being analyzed
`nproc`	Number of cores to use
`seq`	Column name for sequence alignment
`v_call`	Column name for V gene segment gene call
`d_call`	Column name for D gene segment gene call
`j_call`	Column name for J gene segment gene call
`amino_acid`	Perform reconstruction on amino acid sequence (experimental)
`id`	Column name for sequence ID
`clone`	Column name for clone ID
`v_germ_start`	Column name of index of V segment start within germline
`v_germ_end`	Column name of index of V segment end within germline
`v_germ_length`	Column name of index of V segment length within germline
`d_germ_start`	Column name of index of D segment start within germline
`d_germ_end`	Column name of index of D segment end within germline
`d_germ_length`	Column name of index of D segment length within germline

j_germ_start	Column name of index of J segment start within germline
j_germ_end	Column name of index of J segment end within germline
j_germ_length	Column name of index of J segment length within germline
np1_length	Column name in receptor specifying np1 segment length
np2_length	Column name in receptor specifying np2 segment length
na.rm	Remove clones with failed germline reconstruction?
fields	Character vector of additional columns to use for grouping. Sequences with disjoint values in the specified fields will be considered as separate clones.
verbose	amount of rubbish to print
...	Additional arguments passed to buildGermline

Details

Return object adds/edits following columns:

- seq: Sequences potentially padded same length as germline
- germline_alignment: Full length germline
- germline_alignment_d_mask: Full length, D region masked
- vonly: V gene segment of germline if vonly=TRUE
- regions: String of VDJ segment in position if use_regions=TRUE

Value

Tibble with reconstructed germlines

See Also

createGermlines buildGermline, stitchVDJ

Examples

```
vdj_dir <- system.file("extdata", "germlines", "imgt", "human", "vdj", package="dowser")
imgt <- readIMGT(vdj_dir)
db <- createGermlines(ExampleAirr[1,], imgt)
```

downsampleClone downsampleClone *Down-sample clone to maximum tip/switch ratio*

Description

downsampleClone Down-sample clone to maximum tip/switch ratio

Usage

```
downsampleClone(clone, trait, tip_switch = 20, tree = NULL)
```

Arguments

clone	an airrClone object
trait	trait considered for rarefaction getTrees
tip_switch	maximum tip/switch ratio
tree	a phylo tree object correspond to clone

Value

A vector with sequence for each locus at a specified node in tree.

dowser *The dowser package*

Description

dowser is a phylogenetic analysis package as part of the Immcantation suite of tools. For additional details regarding the use of the dowser package see the vignettes:
`browseVignettes("dowser")`

References

1. Hoehn, KB, Pybus, OG, Kleinstein SH (2020) Phylogenetic analysis of migration, differentiation, and class switching in B cells. https://www.biorxiv.org/content/10.1101/2020.05.30.124446v1

`ExampleAirr` *Example AIRR database*

Description

A small example database subset from Laserson and Vigneault et al, 2014.

Usage

```
ExampleAirr
```

Format

A data.frame with the following AIRR style columns:

- `sequence_id`: Sequence identifier
- `sequence_alignment`: IMGT-gapped observed sequence.
- `germline_alignment_d_mask`: IMGT-gapped germline sequence with N, P and D regions masked.
- `v_call`: V region allele assignments.
- `v_call_genotyped`: TIgGER corrected V region allele assignment.
- `d_call`: D region allele assignments.
- `j_call`: J region allele assignments.
- `junction`: Junction region sequence.
- `junction_length`: Length of the junction region in nucleotides.
- `np1_length`: Combined length of the N and P regions proximal to the V region.
- `np2_length`: Combined length of the N and P regions proximal to the J region.
- `sample`: Sample identifier. Time in relation to vaccination.
- `isotype`: Isotype assignment.
- `duplicate_count`: Copy count (number of duplicates) of the sequence.
- `clone_id`: Change-O assignment clonal group identifier.

References

1. Laserson U and Vigneault F, et al. High-resolution antibody dynamics of vaccine-induced immune responses. Proc Natl Acad Sci USA. 2014 111:4928-33.

See Also

ExampleDbChangeo ExampleClones

ExampleClones *Example Ig lineage trees*

Description

A tibble of Ig lineage trees generated from the `ExampleAirr` file

Usage

```
ExampleClones
```

Format

A tibble of airrClone and phylo objects output by getTrees.

- `clone_id`: Clonal cluster
- `data`: List of airrClone objects
- `seqs`: Number of sequences
- `trees`: List of phylo objects

See Also

ExampleClones

ExampleDbChangeo *Example Change-O database*

Description

A small example database subset from Laserson and Vigneault et al, 2014.

Usage

```
ExampleDbChangeo
```

Format

A data.frame with the following Change-O style columns:

- `SEQUENCE_ID`: Sequence identifier
- `SEQUENCE_IMGT`: IMGT-gapped observed sequence.
- `GERMLINE_IMGT_D_MASK`: IMGT-gapped germline sequence with N, P and D regions masked.
- `V_CALL`: V region allele assignments.
- `V_CALL_GENOTYPED`: TIgGER corrected V region allele assignment.

- D_CALL: D region allele assignments.
- J_CALL: J region allele assignments.
- JUNCTION: Junction region sequence.
- JUNCTION_LENGTH: Length of the junction region in nucleotides.
- NP1_LENGTH: Combined length of the N and P regions proximal to the V region.
- NP2_LENGTH: Combined length of the N and P regions proximal to the J region.
- SAMPLE: Sample identifier. Time in relation to vaccination.
- ISOTYPE: Isotype assignment.
- DUPCOUNT: Copy count (number of duplicates) of the sequence.
- CLONE: Change-O assignment clonal group identifier.

References

1. Laserson U and Vigneault F, et al. High-resolution antibody dynamics of vaccine-induced immune responses. Proc Natl Acad Sci USA. 2014 111:4928-33.

See Also

ExampleAirr ExampleClones

`findSwitches` *Create a bootstrap distribution for clone sequence alignments, and estimate trees for each bootstrap replicate.*

Description

`findSwitches` Phylogenetic bootstrap function.

Usage

```
findSwitches(
  clones,
  permutations,
  trait,
  igphyml,
  fixtrees = FALSE,
  downsample = TRUE,
  tip_switch = 20,
  nproc = 1,
  dir = NULL,
  id = NULL,
  modelfile = NULL,
  build = "pratchet",
  exec = NULL,
  quiet = 0,
```

```
  rm_temp = TRUE,
  palette = NULL,
  resolve = 2,
  rep = NULL,
  keeptrees = FALSE,
  lfile = NULL,
  seq = NULL,
  boot_part = "locus",
  force_resolve = FALSE,
  ...
)
```

Arguments

clones	tibble airrClone objects, the output of formatClones
permutations	number of bootstrap replicates to perform
trait	trait to use for parsimony models
igphyml	location of igphyml executible
fixtrees	keep tree topologies fixed? (bootstrapping will not be perfomed)
downsample	downsample clones to have a maximum specified tip/switch ratio?
tip_switch	maximum allowed tip/switch ratio if downsample=TRUE
nproc	number of cores to parallelize computations
dir	directory where temporary files will be placed (required if igphyml or dnapars specified)
id	unique identifer for this analysis (required if igphyml or dnapars specified)
modelfile	file specifying parsimony model to use
build	program to use for tree building (phangorn, dnapars)
exec	location of desired phylogenetic executable
quiet	amount of rubbish to print to console
rm_temp	remove temporary files (default=TRUE)
palette	a named vector specifying colors for each state
resolve	how should polytomies be resolved? 0=none, 1=max parsminy, 2=max ambiguity + polytomy skipping, 3=max ambiguity
rep	current bootstrap replicate (experimental)
keeptrees	keep trees estimated from bootstrap replicates? (TRUE)
lfile	lineage file input to igphyml if desired (experimental)
seq	column name containing sequence information
boot_part	is "locus" bootstrap columns for each locus separately
force_resolve	continue even if polytomy resolution fails?
...	additional arguments to be passed to tree building program

Details

Tree building details are the same as getTrees. If `keeptrees=TRUE` (default) the returned object will contain a list named "trees" which contains a list of estimated tree objects for each bootstrap replicate. The object is structured like: trees[[<replicate>]][[<tree index>]]. If `igphyml` is specified (as well as `trait`), the returned object will contain a `tibble` named "switches" containing switch count information. This object can be passed to testSP and other functions to perform parsimony based trait value tests.

Value

A list of trees and/or switch counts for each bootstrap replicate.

See Also

Uses output from formatClones with similar arguments to getTrees. Output can be visualized with plotTrees, and tested with testPS, testSC, and testSP.

Examples

```
## Not run:
data(ExampleAirr)
ExampleAirr$sample_id <- sample(ExampleAirr$sample_id)
clones <- formatClones(ExampleAirr, trait="sample_id")

igphyml <- "~/apps/igphyml/src/igphyml"
btrees <- findSwitches(clones[1:2], permutations=10, nproc=1,
   igphyml=igphyml, trait="sample_id")
plotTrees(btrees$trees[[4]])[[1]]
testPS(btrees$switches)

## End(Not run)
```

`formatClones` *Generate an ordered list of airrClone objects for lineage construction*

Description

`formatClones` takes a `data.frame` or `tibble` with AIRR or Change-O style columns as input and masks gap positions, masks ragged ends, removes duplicates sequences, and merges annotations associated with duplicate sequences. If specified, it will un-merge duplicate sequences with different values specified in the `trait` option. It returns a list of `airrClone` objects ordered by number of sequences which serve as input for lineage reconstruction.

Usage

```
formatClones(
  data,
  seq = "sequence_alignment",
  clone = "clone_id",
  subclone = "subclone_id",
  nproc = 1,
  chain = "H",
  heavy = "IGH",
  cell = "cell_id",
  locus = "locus",
  minseq = 2,
  split_light = FALSE,
  majoronly = FALSE,
  columns = NULL,
  ...
)
```

Arguments

data	data.frame containing the AIRR or Change-O data for a clone. See makeAirrClone for required columns and their defaults
seq	sequence alignment column name.
clone	name of the column containing the identifier for the clone. All entries in this column should be identical.
subclone	name of the column containing the identifier for the subclone.
nproc	number of cores to parallelize formating over.
chain	if HL, include light chain information if available.
heavy	name of heavy chain locus (default = "IGH")
cell	name of the column containing cell assignment information
locus	name of the column containing locus information
minseq	minimum numbner of sequences per clone
split_light	split or lump subclones? See getSubclones.
majoronly	only return largest subclone and sequences without light chains
columns	additional data columns to include in output
...	additional arguments to pass to makeAirrClone

Details

This function is a wrapper for makeAirrClone. Also removes whitespace, ;, :, and = from ids

Value

A tibble of airrClone objects containing modified clones.

See Also

Executes in order makeAirrClone. Returns a tibble of airrClone objects which serve as input to getTrees and findSwitches.

Examples

```
data(ExampleAirr)
# Select two clones, for demonstration purpose
sel <- c("3170", "3184")
clones <- formatClones(ExampleAirr[ExampleAirr$clone_id %in% sel,],trait="sample_id")
```

`getDivergence` *Get divergence from root of tree for each tip*

Description

`getDivergence` get sum of branch lengths leading from the root of the tree. If the germline sequence is included in the tree, this will equal the germline divergence. If germline removed, this will equal the MRCA divergence

Usage

```
getDivergence(phy, minlength = 0.001)
```

Arguments

`phy`	Tree object
`minlength`	Branch lengths to collapse in trees

Value

A named vector of each tip's divergence from the tree's root.

`getGermline` *getGermline get germline segment from specified receptor and segment*

Description

getGermline get germline segment from specified receptor and segment

Usage

```
getGermline(
  receptor,
  references,
  segment,
  field,
  germ_start,
  germ_end,
  germ_length,
  germ_aa_start,
  germ_aa_length,
  amino_acid = FALSE
)
```

Arguments

receptor	row from AIRR-table containing sequence of interest
references	list of reference segments. Must be specific to organism, locus, and segment
segment	Gene segment to search. Must be V, D, or J.
field	Column name for segment gene call (e.g. v_call)
germ_start	Column name of index of segment start within germline segment (e.g. v_germline_start)
germ_end	Similar to germ_start, but specifies end of segment (e.g. v_germline_end)
germ_length	Similar to germ_start, but specifies length of segment (e.g. v_germline_end)
germ_aa_start	Column name of index of segment start within germline segment in AA (if amino_acid=TRUE, e.g. v_germline_start)
germ_aa_length	Similar to germ_start, but specifies length of segment in AA (if amino_acid=TRUE, e.g. v_germline_end)
amino_acid	Perform reconstruction on amino acid sequence (experimental)

Value

String of germline sequence from specified segment aligned with the sequence in the seq column of receptor.

---

getNodeSeq *Return IMGT gapped sequence of specified tree node*

Description

getNodeSeq Sequence retrieval function.

Usage

```
getNodeSeq(data, node, tree = NULL, clone = NULL, gaps = TRUE)
```

Arguments

data	a tibble of airrClone objects, the output of getTrees
node	numeric node in tree (see details)
tree	a phylo tree object containing node
clone	if tree not specified, supply clone ID in data
gaps	add IMGT gaps to output sequences?

Details

Use plotTrees(trees)[[1]] + geom_label(aes(label=node))+geom_tippoint() to show node labels, and getNodeSeq to return internal node sequences

Value

A vector with sequence for each locus at a specified node in tree.

See Also

getTrees

---

getPalette *Get a color palette for a predefined set of trait values*

Description

getPalette Gets a color palette for a predefined set of trait values

Usage

```
getPalette(states, palette)
```

Arguments

states	states in model
palette	The colorbrewer palette to use

Value

A named vector with each state corresponding to a color

See Also

getTrees, plotTrees

getSeq *Deprecated! Use getNodeSeq*

Description

getSeq Sequence retrieval function.

Usage

```
getSeq(data, node, tree = NULL, clone = NULL, gaps = TRUE)
```

Arguments

data	a tibble of airrClone objects, the output of getTrees
node	numeric node in tree (see details)
tree	a phylo tree object containing node
clone	if tree not specified, supply clone ID in data
gaps	add IMGT gaps to output sequences?

Value

A vector with sequence for each locus at a specified node in tree.

See Also

getTrees

getSubclones *Define subclones based on light chain rearrangements*

Description

getSubclones plots a tree or group of trees

Usage

```
getSubclones(
  heavy,
  light,
  nproc = 1,
  minseq = 1,
  id = "sequence_id",
  seq = "sequence_alignment",
  clone = "clone_id",
  cell = "cell_id",
```

```
  v_call = "v_call",
  j_call = "j_call",
  junc_len = "junction_length",
  nolight = "missing"
)
```

Arguments

heavy	a tibble containing heavy chain sequences with clone_id
light	a tibble containing light chain sequences
nproc	number of cores for parallelization
minseq	minimum number of sequences per clone
id	name of the column containing sequence identifiers.
seq	name of the column containing observed DNA sequences. All sequences in this column must be multiple aligned.
clone	name of the column containing the identifier for the clone. All entries in this column should be identical.
cell	name of the column containing identifier for cells.
v_call	name of the column containing V-segment allele assignments. All entries in this column should be identical to the gene level.
j_call	name of the column containing J-segment allele assignments. All entries in this column should be identical to the gene level.
junc_len	name of the column containing the length of the junction as a numeric value. All entries in this column should be identical for any given clone.
nolight	string to use to indicate a missing light chain

Details

1. Make temporary array containing light chain clones 2. Enumerate all possible V and J combinations 3. Determine which combination is the most frequent 4. Assign sequences with that combination to clone t 5. Copy those sequences to return array 6. Remove all cells with that combination from temp array 7. Repeat 1-5 until temporary array zero. If there is more than rearrangement with the same V/J in the same cell, pick the one with the highest non-ambiguous characters.

Value

a tibble containing

`getTrees` *Estimate lineage tree topologies, branch lengths, and internal node states if desired*

Description

`getTrees` Tree building function.

Usage

```
getTrees(
  clones,
  trait = NULL,
  id = NULL,
  dir = NULL,
  modelfile = NULL,
  build = "pratchet",
  exec = NULL,
  igphyml = NULL,
  fixtrees = FALSE,
  nproc = 1,
  quiet = 0,
  rm_temp = TRUE,
  palette = NULL,
  seq = NULL,
  collapse = FALSE,
  ...
)
```

Arguments

`clones`	a tibble of `airrClone` objects, the output of formatClones
`trait`	trait to use for parsimony models (required if `igphyml` specified)
`id`	unique identifer for this analysis (required if `igphyml` or `dnapars` specified)
`dir`	directory where temporary files will be placed.
`modelfile`	file specifying parsimony model to use
`build`	program to use for tree building (pratchet, pml, dnapars, dnaml, igphyml)
`exec`	location of desired phylogenetic executable
`igphyml`	optional location of igphyml executible for parsimony
`fixtrees`	if TRUE, use supplied tree topologies
`nproc`	number of cores to parallelize computations
`quiet`	amount of rubbish to print to console
`rm_temp`	remove temporary files (default=TRUE)

palette	a named vector specifying colors for each state
seq	column name containing sequence information
collapse	Collapse internal nodes with identical sequences?
...	Additional arguments passed to tree building programs

Details

Estimates phylogenetic tree topologies and branch lengths for a list of airrClone objects. By default, it will use phangnorn::pratchet to estimate maximum parsimony tree topologies, and ape::acctran to estimate branch lengths. If igpyhml is specified, internal node trait values will be predicted by maximum parsimony. In this case, dir will need to be specified as a temporary directory to place all the intermediate files (will be created if not available). Further, id will need to specified to serve as a unique identifier for the temporary files. This should be chosen to ensure that multiple getTrees calls using the same dir do not overwrite each others files.

modelfile is written automatically if not specified, but doesn't include any constraints. Intermediate files are deleted by default. This can be toggled using (rm_files).

For examples and vignettes, see https://dowser.readthedocs.io

Value

A list of phylo objects in the same order as data.

See Also

formatClones, findSwitches, buildPhylo, buildPratchet, buildPML, buildIgphyml

Examples

```
data(ExampleClones)

trees <- getTrees(ExampleClones[10,])
plotTrees(trees)[[1]]

## Not run:
data(ExampleClones)

trees <- getTrees(ExampleClones[10,],igphyml="/path/to/igphyml",
         id="temp",dir="temp", trait="sample_id")
plotTrees(trees)[[1]]

## End(Not run)
```

makeAirrClone *Generate a airrClone object for lineage construction*

Description

makeAirrClone takes a data.frame with AIRR or Change-O style columns as input and masks gap positions, masks ragged ends, removes duplicates sequences, and merges annotations associated with duplicate sequences. It returns a airrClone object which serves as input for lineage reconstruction.

Usage

```
makeAirrClone(
  data,
  id = "sequence_id",
  seq = "sequence_alignment",
  germ = "germline_alignment_d_mask",
  v_call = "v_call",
  j_call = "j_call",
  junc_len = "junction_length",
  clone = "clone_id",
  mask_char = "N",
  max_mask = 0,
  pad_end = TRUE,
  text_fields = NULL,
  num_fields = NULL,
  seq_fields = NULL,
  add_count = TRUE,
  verbose = FALSE,
  collapse = TRUE,
  chain = "H",
  heavy = NULL,
  cell = "cell_id",
  locus = "locus",
  traits = NULL,
  mod3 = TRUE,
  randomize = TRUE,
  use_regions = TRUE,
  dup_singles = FALSE
)
```

Arguments

data	data.frame containing the AIRR or Change-O data for a clone. See Details for the list of required columns and their default values.
id	name of the column containing sequence identifiers.

seq	name of the column containing observed DNA sequences. All sequences in this column must be multiple aligned.
germ	name of the column containing germline DNA sequences. All entries in this column should be identical for any given clone, and they must be multiple aligned with the data in the seq column.
v_call	name of the column containing V-segment allele assignments. All entries in this column should be identical to the gene level.
j_call	name of the column containing J-segment allele assignments. All entries in this column should be identical to the gene level.
junc_len	name of the column containing the length of the junction as a numeric value. All entries in this column should be identical for any given clone.
clone	name of the column containing the identifier for the clone. All entries in this column should be identical.
mask_char	character to use for masking and padding.
max_mask	maximum number of characters to mask at the leading and trailing sequence ends. If NULL then the upper masking bound will be automatically determined from the maximum number of observed leading or trailing Ns amongst all sequences. If set to 0 (default) then masking will not be performed.
pad_end	if TRUE pad the end of each sequence with mask_char to make every sequence the same length.
text_fields	text annotation columns to retain and merge during duplicate removal.
num_fields	numeric annotation columns to retain and sum during duplicate removal.
seq_fields	sequence annotation columns to retain and collapse during duplicate removal. Note, this is distinct from the seq and germ arguments, which contain the primary sequence data for the clone and should not be repeated in this argument.
add_count	if TRUE add an additional annotation column called COLLAPSE_COUNT during duplicate removal that indicates the number of sequences that were collapsed.
verbose	passed on to collapseDuplicates. If TRUE, report the numbers of input, discarded and output sequences; otherwise, process sequences silently.
collapse	collapse identical sequences?
chain	if HL, include light chain information if available.
heavy	name of heavy chain locus (default = "IGH")
cell	name of the column containing cell assignment information
locus	name of the column containing locus information
traits	column ids to keep distinct during sequence collapse
mod3	pad sequences to length mutliple three?
randomize	randomize sequence order? Important if using PHYLIP
use_regions	assign CDR/FWR regions?
dup_singles	Duplicate sequences in singleton clones to include them as trees?

Details

The input data.frame (data) must columns for each of the required column name arguments: id, seq, germ, v_call, j_call, junc_len, and clone. Additional annotation columns specified in the traits, text_fields, num_fields or seq_fields arguments will be retained in the data slot of the return object, but are not required. These options differ by their behavior among collapsed sequences. Identical sequences that differ by any values specified in the traits option will be kept distinct. Identical sequences that differ only by values in the num_fields option will be collapsed and the values of their num_fields columns will be added together. Similar behavior occurs with text_fields but the unique values will concatenated with a comma.

The default columns are IMGT-gapped sequence columns, but this is not a requirement. However, all sequences (both observed and germline) must be multiple aligned using some scheme for both proper duplicate removal and lineage reconstruction.

The value for the germline sequence, V-segment gene call, J-segment gene call, junction length, and clone identifier are determined from the first entry in the germ, v_call, j_call, junc_len and clone columns, respectively. For any given clone, each value in these columns should be identical.

To allow for cases where heavy and light chains are used, this function returns three sequence columns for heavy chains (sequence), light chain (lsequence, empty if none available), and concatenated heavy+light chain (hlsequence). These contain sequences in alignment with germline, lgermline, and hlgermline slots, respectively. The sequence column used for build trees is specified in the phylo_seq slot. Importantly, this column is also the sequence column that also has uninformative columns removed by cleanAlignment. It is highly likely we will change this system to a single sequence and germline slot in the near future.

The airrClone object also contains vectors locus, region, and numbers, which contain the locus, IMGT region, and IMGT number for each position in the sequence column specified in phylo_seq. If IMGT-gapped sequences are not supplied, this will likely result in an error. Specify use_regions=FALSE if not using IMGT-gapped sequences

Value

A airrClone object containing the modified clone.

See Also

Returns an airrClone. See formatClones to generate an ordered list of airrClone objects.

Examples

```
data(ExampleAirr)
airr_clone <- makeAirrClone(ExampleAirr[ExampleAirr$clone_id=="3184",])
```

---

makeModelFile *Make a parsimony model file*

Description

makeModelFile Filler

Usage

```
makeModelFile(file, states, constraints = NULL)
```

Arguments

file	model file name to write.
states	vector of states to include in model.
constraints	constraints to add to model.

Details

Currently the only option for constraints is "irrev", which forbids switches moving from left to right in the states vector.

Value

Name of model file

See Also

readModelFile, getTrees, findSwitches

---

maskCodons maskCodons *Masks codons split by insertions*

Description

maskCodons Masks codons split by insertions

Usage

```
maskCodons(
  id,
  q,
  s,
  keep_alignment = FALSE,
  gap_opening = 5,
  gap_extension = 1,
  keep_insertions = FALSE,
  mask = TRUE
)
```

Arguments

id	sequence id
q	(query) un-aligned input sequence (sequence)
s	(subject) aligned input sequence (sequence_alignment)
keep_alignment	store q and s alignments
gap_opening	gap opening penalty (Biostrings::pairwiseALignment)
gap_extension	gap extension penalty (Biostrings::pairwiseALignment)
keep_insertions	return removed insertion sequences?
mask	if FALSE, don't mask codons

Details

Performs global alignment of q and s, masks codons in s that are split by insertions (see example) masking_note notes codon positions in subject_alignment sequence that were masked, if found. subject_alignment contains subject sequence aligned to query (q) sequence query_alignment contains query sequence aligned to subject (q) sequence sequence_masked will be NA if frameshift or alignment error detected/

Value

A list with split codons masked, if found (sequence_masked).

See Also

maskSequences, Biostrings::pairwiseAlignment.

Examples

```
s = "ATCATCATC..."
q = "ATCTTTATCATC"
print(maskCodons(1,q,s,TRUE))

s <- "ATCATCATC..."
q <- "ATTTTCATCATC"
print(maskCodons("test",q,s,keep_alignment=TRUE,keep_insertions=TRUE))
```

---

maskSequences maskSequences *Mask codons split by insertions in V gene*

Description

maskSequences Mask codons split by insertions in V gene

Usage

```
maskSequences(
  data,
  sequence_id = "sequence_id",
  sequence = "sequence",
  sequence_alignment = "sequence_alignment",
  v_sequence_start = "v_sequence_start",
  v_sequence_end = "v_sequence_end",
  v_germline_start = "v_germline_start",
  v_germline_end = "v_germline_end",
  junction_length = "junction_length",
  keep_alignment = FALSE,
  keep_insertions = FALSE,
  mask_codons = TRUE,
  mask_cdr3 = TRUE,
  nproc = 1
)
```

Arguments

data	BCR data table
sequence_id	sequence id column
sequence	input sequence column (query)
sequence_alignment	aligned (IMGT-gapped) sequence column (subject)
v_sequence_start	V gene start position in sequence
v_sequence_end	V gene end position in sequence
v_germline_start	V gene start position in sequence_alignment
v_germline_end	V gene end position in sequence_alignment
junction_length	name of junction_length column
keep_alignment	store alignment of query and subject sequences?
keep_insertions	return removed insertion sequences?
mask_codons	mask split codons?
mask_cdr3	mask CDR3 sequences?
nproc	number of cores to use

Details

Performs global alignment of sequence and sequence_alignment, masking codons in sequence_alignment that are split by insertions (see examples) masking_note notes codon positions in subject_alignment sequence that were masked, if found. subject_alignment contains subject sequence aligned to query

sequence (only if keep_alignment=TRUE) query_alignment contains query sequence aligned to subject sequence (only if keep_alignment=TRUE) sequence_masked will be NA if frameshift or alignment error detected. This will be noted insertions column will be returned if keep_insertions=TRUE, contains a comma-separated list of each <position in query alignment>-<sequence>. See example. in masking_note.

Value

A tibble with masked sequence in sequence_masked column, as well as other columns.

See Also

maskCodons, Biostrings::pairwiseAlignment.

`plotTrees` *Plot a tree with colored internal node labels using ggtree*

Description

`plotTrees` plots a tree or group of trees

Usage

```
plotTrees(
  trees,
  nodes = FALSE,
  tips = NULL,
  tipsize = NULL,
  scale = 0.01,
  node_palette = "Dark2",
  tip_palette = node_palette,
  base = FALSE,
  layout = "rectangular",
  node_nums = FALSE,
  tip_nums = FALSE,
  title = TRUE,
  labelsize = NULL,
  common_scale = FALSE
)
```

Arguments

`trees`	A tibble containing `phylo` and `airrClone` objects
`nodes`	color internal nodes if possible?
`tips`	color tips if possible?
`tipsize`	size of tip shape objects

scale	width of branch length scale bar
node_palette	color palette for nodes
tip_palette	color palette for tips
base	recursion base case (don't edit)
layout	rectangular or circular tree layout?
node_nums	plot internal node numbers?
tip_nums	plot tip numbers?
title	use clone id as title?
labelsize	text size
common_scale	strecth plots so branches are on same scale? determined by sequence with highest divergence

Details

Function uses ggtree functions to plot tree topologlies estimated by getTrees, and findSwitches.
Object can be further modified with ggtree functions. Please check out https://bioconductor.org/packages/devel/bioc/vignette
and cite ggtree in addition to dowser if you use this function.

Value

a grob containing a tree plotted by ggtree.

See Also

getTrees, findSwitches

Examples

```
data(ExampleClones)
trees <- getTrees(ExampleClones[10,])
plotTrees(trees)[[1]]
```

---

readFasta *Read a fasta file into a list of sequences* readFasta *reads a fasta file*

Description

Read a fasta file into a list of sequences readFasta reads a fasta file

Usage

```
readFasta(file)
```

Arguments

file	FASTA file

Value

List of sequences

---

readIMGT readIMGT *read in IMGT database*

Description

Loads all reference germlines from an Immcantation-formatted IMGT database.

Usage

```
readIMGT(dir, quiet = FALSE)
```

Arguments

dir	directory containing Immcantation-formatted IMGT database
quiet	print warnings?

Details

Input directory must be formatted to Immcantation standard. See https://changeo.readthedocs.io/en/stable/examples/igblast.h
for example of how to download.

Value

List of lists, leading to IMGT-gapped nucleotide sequences. Structure of object is list[[organism]][[locus]][[segment]]
Organism refers to species (i.e. human, mouse) locus refers to locus (e.g. IGH, IGK, TRA) segment
refers to gene segment caegory (V, D, or J)

Examples

```
# vdj_dir contains a minimal example of reference germlines
# (IGHV3-11*05, IGHD3-10*01 and IGHJ5*02)
# which are the gene assignments for ExamapleDb[1,]
vdj_dir <- system.file("extdata", "germlines", "imgt", "human", "vdj", package="dowser")
imgt <- readIMGT(vdj_dir)
```

`readLineages` *Read in all trees from a lineages file*

Description

Read in all trees from a lineages file

Usage

```
readLineages(
  file,
  states = NULL,
  palette = "Dark2",
  run_id = "",
  quiet = TRUE,
  append = NULL,
  format = "nexus",
  type = "jointpars"
)
```

Arguments

`file`	IgPhyML lineage file
`states`	states in parsimony model
`palette`	palette for coloring internal nodes
`run_id`	id used for IgPhyML run
`quiet`	avoid printing rubbish on screen?
`append`	string appended to fasta files
`format`	format of input file with trees
`type`	Read in parsimony reconstructions or ancestral sequence reconstructions? "jointpars" reads in parsimony states, others read in sequences in internal nodes

Value

A list of phylo objects from `file`.

readModelFile *Read in a parsimony model file*

Description

`readModelFile` Filler

Usage

```
readModelFile(file, useambig = FALSE)
```

Arguments

`file`	parimony model file.
`useambig`	use ambiguous naming as specified in the file?

Value

A named vector containing the states of the model

See Also

makeModelFile, findSwitches, getTrees

reconIgPhyML *Do IgPhyML maximum parsimony reconstruction*

Description

`reconIgPhyML` IgPhyML parsimony reconstruction function

Usage

```
reconIgPhyML(
  file,
  modelfile,
  id,
  igphyml = "igphyml",
  mode = "switches",
  type = "recon",
  nproc = 1,
  quiet = 0,
  rm_files = FALSE,
  rm_dir = NULL,
  states = NULL,
```

```
  palette = NULL,
  resolve = 2,
  rseed = NULL,
  force_resolve = FALSE,
  ...
)
```

Arguments

file	IgPhyML lineage file (see writeLineageFile)
modelfile	File specifying parsimony model
id	id for IgPhyML run
igphyml	location of igphyml executable
mode	return trees or count switches? (switches or trees)
type	get observed switches or permuted switches?
nproc	cores to use for parallelization
quiet	amount of rubbish to print
rm_files	remove temporary files?
rm_dir	remove temporary directory?
states	states in parsimony model
palette	palette for coloring tree (see getPallete)
resolve	level of polytomy resolution. 0=none, 1=maximum parsimony, 2=maximum ambiguity
rseed	random number seed if desired
force_resolve	continue even if polytomy resolution fails?
...	additional arguments

Value

Either a tibble of switch counts or a list of trees with internal nodes predicted by parsimony.

---

rerootTree *Reroot phylogenetic tree to have its germline sequence at a zero-length branch to a node which is the direct ancestor of the tree's UCA. Assigns* uca *to be the ancestral node to the tree's germline sequence, as* germid *as the tree's germline sequence ID.*

Description

Reroot phylogenetic tree to have its germline sequence at a zero-length branch to a node which is the direct ancestor of the tree's UCA. Assigns uca to be the ancestral node to the tree's germline sequence, as germid as the tree's germline sequence ID.

Usage

```
rerootTree(tree, germline, min = 0.001, verbose = 1)
```

Arguments

tree	An ape phylo object
germline	ID of the tree's predicted germline sequence
min	Maximum allowed branch length from germline to root
verbose	amount of rubbish to print

Value

phylo object rooted at the specified germline

---

resolvePolytomies *Resolve polytomies to have the minimum number of single timepoint clades*

Description

Resolve polytomies to have the minimum number of single timepoint clades

Usage

```
resolvePolytomies(
  phy,
  clone,
  minlength = 0.001,
  time = "time",
  sequence = "sequence_id",
  germline = "Germline",
  verbose = FALSE
)
```

Arguments

phy	Tree object
clone	airrClone data object corresponding to phy
minlength	Branch lengths to collapse in trees
time	Column name holding numeric time information
sequence	Column name holding sequence ID
germline	Germline sequence name
verbose	Print lots of rubbish while running?

Details

Iteratively identifies polytomies (clusters of < minlength branches), prunes each descendant branch, combines clades with the same timepoint before grouping them back together. Checks to make sure that the divergence of each tip is the same after resolution.

Value

A phylo tree object in which polytomies are resolved to have the minimum number of single timepoint clades.

See Also

Uses output from getTrees during correlationTest.

`runCorrelationTest` *Run correlationTest, based on https://doi.org/10.1111/2041-210X.12466*

Description

`runCorrelationTest` performs root-to-tip regression permutation test

Usage

```
runCorrelationTest(
  phy,
  clone,
  permutations,
  minlength = 0.001,
  polyresolve = TRUE,
  permutation = c("clustered", "uniform"),
  time = "time",
  sequence = "sequence_id",
  germline = "Germline",
  verbose = TRUE,
  alternative = c("greater", "two.sided")
)
```

Arguments

`phy`	Tree object
`clone`	airrClone data object corresponding to phy
`permutations`	Number of permutations to run
`minlength`	Branch lengths to collapse in trees
`polyresolve`	Resolve polytomies to have a minimum number of single timepoint clades
`permutation`	Permute among single timepoint clades or uniformly among tips

time	Column name holding numeric time information
sequence	Column name holding sequence ID
germline	Germline sequence name
verbose	Print lots of rubbish while running?
alternative	Is alternative that the randomized correlation are greater than or equal to observed, or greater/less than?

Details

See correlationTest for details

Value

A list of statistics from running the permutation test.

See Also

correlationTest.

`scaleBranches` *Scale branch lengths to represent either mutations or mutations per site.*

Description

`scaleBranches` Branch length scaling function.

Usage

```
scaleBranches(clones, edge_type = "mutations")
```

Arguments

clones	a tibble of `airrClone` and `phylo` objects, the output of getTrees.
edge_type	Either `genetic_distance` (mutations per site) or `mutations`

Details

Uses clones$trees[[1]]$edge_type to determine how branches are currently scaled.

Value

A tibble with `phylo` objects that have had branch lengths rescaled as specified.

See Also

getTrees

stitchRegions *stitchRegions Similar to stitchVDJ but with segment IDs instead of nulecotides*

Description

stitchRegions Similar to stitchVDJ but with segment IDs instead of nulecotides

Usage

```
stitchRegions(
  receptor,
  v_seq,
  d_seq,
  j_seq,
  np1_length = "np1_length",
  np2_length = "np1_length",
  n1_length = "n1_length",
  p3v_length = "p3v_length",
  p5d_length = "p5d_length",
  p3d_length = "p3d_length",
  n2_length = "n2_length",
  p5j_length = "p5j_length",
  np1_aa_length = "np1_aa_length",
  np2_aa_length = "np2_aa_length",
  amino_acid = FALSE
)
```

Arguments

receptor	row from AIRR-table containing sequence of interest
v_seq	germline V segment sequence from getGermline
d_seq	germline D segment sequence from getGermline
j_seq	germline J segment sequence from getGermline
np1_length	Column name in receptor specifying np1 segment length (e.g. np1_length)
np2_length	Column name in receptor specifying np2 segment length (e.g. np1_length)
n1_length	Column name in receptor specifying n1 segment length (experimental)
p3v_length	Column name in receptor specifying p3v segment length (experimental)
p5d_length	Column name in receptor specifying p5d segment length (experimental)
p3d_length	Column name in receptor specifying p3d segment length (experimental)
n2_length	Column name in receptor specifying n2 segment length (experimental)
p5j_length	Column name in receptor specifying p5j segment length (experimental)
np1_aa_length	Column name in receptor specifying np1 segment length in AA (if amino_acid=TRUE, e.g. np1_length)

np2_aa_length	Column name in receptor specifying np2 segment length in AA (if amino_acid=TRUE, e.g. np1_length)
amino_acid	Perform reconstruction on amino acid sequence (experimental)

Value

Full length germline VDJ sequence with segment IDs instead of nucleotides.

See Also

stitchVDJ

---

stitchVDJ *stitchVDJ combines germline gene segments to a single string*

Description

stitchVDJ combines germline gene segments to a single string

Usage

```
stitchVDJ(
  receptor,
  v_seq,
  d_seq,
  j_seq,
  np1_length = "np1_length",
  np2_length = "np2_length",
  np1_aa_length = "np1_aa_length",
  np2_aa_length = "np2_aa_length",
  amino_acid = FALSE
)
```

Arguments

receptor	row from AIRR-table containing sequence of interest
v_seq	germline V segment sequence from getGermline
d_seq	germline D segment sequence from getGermline
j_seq	germline J segment sequence from getGermline
np1_length	Column name in receptor specifying np1 segment length (e.g. np1_length)
np2_length	Column name in receptor specifying np2 segment length (e.g. np1_length)
np1_aa_length	Column name in receptor specifying np1 segment length in AA (if amino_acid=TRUE, e.g. np1_length)
np2_aa_length	Column name in receptor specifying np2 segment length in AA (if amino_acid=TRUE, e.g. np1_length)
amino_acid	Perform reconstruction on amino acid sequence (experimental)

Value

Full length germline VDJ sequence aligned with aligned with the sequence in the `seq` column of `receptor`.

`testPS` *Performs PS (parsimony score) test on switch data*

Description

`testPS` performs a PS test

Usage

```
testPS(
  switches,
  bylineage = FALSE,
  pseudocount = 0,
  alternative = c("less", "two.sided", "greater")
)
```

Arguments

`switches`	Data frame from findSwitches
`bylineage`	Perform test for each lineage individually? (FALSE)
`pseudocount`	Pseudocount for P value calculations
`alternative`	Perform one-sided (`greater` or `less`) or `two.sided` test

Details

Output data table columns: RECON = PS for observed data PERMUTE = PS for permuted data DELTA = RECON - PERMUTE PLT = p value for DELTA < 0 PGT = p value for DELTA < 0

- `RECON`: PS for observed data.
- `PERMUTE`: PS for permuted data.
- `DELTA`: RECON - PERMUTE.
- `PLT`: p value that DELTA < 0
- `PGT`: p value that DELTA > 0
- `STAT`: Statistic used (PS).
- `REP`: Bootstrap repetition.
- `REPS`: Total number of ootstrap repetition.

Value

A list containing a `tibble` with mean PS statistics, and another with PS statistics per repetition.

See Also

Uses output from findSwitches. Related to testSP and testSC.

Examples

```
## Not run:
igphyml <- "~/apps/igphyml/src/igphyml"
data(ExampleAirr)
ExampleAirr$sample_id <- sample(ExampleAirr$sample_id)
clones <- formatClones(ExampleAirr, trait="sample_id")
btrees <- findSwitches(clones[1:2], bootstraps=10, nproc=1,
   igphyml=igphyml, trait="sample_id")
testPS(btrees$switches)

## End(Not run)
```

---

testSC *Performs SC (switch count) test on switch data*

Description

testSC performs an SC test

Usage

```
testSC(
  switches,
  dropzeros = TRUE,
  bylineage = FALSE,
  pseudocount = 0,
  from = NULL,
  to = NULL,
  permuteAll = FALSE,
  alternative = c("two.sided", "greater", "less")
)
```

Arguments

switches	Data frame from findSwitches
dropzeros	Drop switches with zero counts?
bylineage	Perform test for each lineage individually?
pseudocount	Pseudocount for P value calculations
from	Include only switches from this state?
to	Include only switches to this state?
permuteAll	Permute among trees?
alternative	Perform one-sided (greater or less) or two.sided test

Details

Output data table columns: RECON = SC for observed data PERMUTE = SC for permuted data DELTA = RECON - PERMUTE PLT = p value for DELTA < 0 PGT = p value for DELTA < 0

- FROM: State going from.
- TO: State going to.
- RECON: SC for observed data.
- PERMUTE: SC for permuted data.
- DELTA: RECON - PERMUTE.
- PLT: p value that DELTA < 0
- PGT: p value that DELTA > 0
- STAT: Statistic used (SC).
- REP: Bootstrap repetition.
- REPS: Total number of ootstrap repetition.

Value

A list containing a tibble with mean SC statistics, and another with SC statistics per repetition.

See Also

Uses output from findSwitches. Related to testPS and testSP.

Examples

```
## Not run:
igphyml <- "~/apps/igphyml/src/igphyml"
data(ExampleAirr)
ExampleAirr$sample_id = sample(ExampleAirr$sample_id)
clones = formatClones(ExampleAirr, trait="sample_id")
btrees = findSwitches(clones[1:2], bootstraps=100, nproc=1,
   igphyml=igphyml, trait="sample_id", id="temp", dir="temp")
testSC(btrees$switches)

## End(Not run)
```

---

testSP *Performs SP (switch proportion) test on switch data*

Description

testSP performs an SP test

Usage

```
testSP(
  switches,
  permuteAll = FALSE,
  from = NULL,
  to = NULL,
  dropzeros = TRUE,
  bylineage = FALSE,
  pseudocount = 0,
  alternative = c("greater", "two.sided", "less"),
  tip_switch = 20,
  exclude = FALSE
)
```

Arguments

switches	Data frame from findSwitches
permuteAll	Permute among trees?
from	Include only switches from this state?
to	Include only switches to this state?
dropzeros	Drop switches with zero counts?
bylineage	Perform test for each lineage individually?
pseudocount	Pseudocount for P value calculations
alternative	Perform one-sided (greater or less) or two.sided test
tip_switch	maximum tip/switch ratio
exclude	exclude clones with tip/switch ratio > tip_switch?

Details

Output data table columns: RECON = SP for observed data PERMUTE = SP for permuted data DELTA = RECON - PERMUTE PLT = p value for DELTA < 0 PGT = p value for DELTA < 0

- FROM: State going from.
- TO: State going to.
- RECON: SP for observed data.
- PERMUTE: SP for permuted data.
- DELTA: RECON - PERMUTE.
- PLT: p value that DELTA < 0
- PGT: p value that DELTA > 0
- STAT: Statistic used (SP).
- REP: Bootstrap repetition.
- REPS: Total number of ootstrap repetition.

Value

A list containing a `tibble` with mean SP statistics, and another with SP statistics per repetition.

See Also

Uses output from findSwitches. Related to testPS and testSC.

Examples

```
## Not run:
igphyml <- "~/apps/igphyml/src/igphyml"
data(ExampleAirr)
ExampleAirr$sample_id = sample(ExampleAirr$sample_id)
clones = formatClones(ExampleAirr, trait="sample_id")
btrees = findSwitches(clones[1:2], bootstraps=10, nproc=1,
   igphyml=igphyml, trait="sample_id")
testSP(btrees$switches)

## End(Not run)
```

`treesToPDF` *Simple function for plotting a lot of trees into a pdf*

Description

`treesToPDF` exports trees to a pdf in an orderly fashion

Usage

```
treesToPDF(plots, file, nrow = 2, ncol = 2, ...)
```

Arguments

`plots`	list of tree plots (from plotTrees)
`file`	output file name
`nrow`	number of rows per page
`ncol`	number of columns per page
`...`	optional arguments passed to grDevices::pdf

Value

a PDF of tree plots

See Also

plotTrees

Examples

```
## Not run:
data(ExampleClones)
trees <- getTrees(ExampleClones[10,])
plots <- plotTrees(trees)
treesToPDF(plots,"test.pdf",width=5,height=6)

## End(Not run)
```

`writeLineageFile` *Write lineage file for IgPhyML use*

Description

Write lineage file for IgPhyML use

Usage

```
writeLineageFile(
  data,
  trees = NULL,
  dir = ".",
  id = "N",
  rep = NULL,
  trait = NULL,
  empty = TRUE,
  partition = "single",
  heavy = "IGH"
)
```

Arguments

`data`	list of `airrClone` objects
`trees`	list of `phylo` objects corresponding to `data`
`dir`	directory to write file
`id`	id used for IgPhyML run
`rep`	bootstrap replicate
`trait`	string appended to sequence id in fasta files
`empty`	output uninformative sequences?
`partition`	how to partition omegas
`heavy`	name of heavy chain locus

Value

Name of created lineage file.

Index