

# Package ‘dowser’

March 26, 2026

**Type** Package

**Version** 2.4.1

**Date** 2026-03-25

**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 (2022) <[doi:10.1371/journal.pcbi.1009885](https://doi.org/10.1371/journal.pcbi.1009885)>,

Hoehn et al (2021) <[doi:10.1101/2021.01.06.425648](https://doi.org/10.1101/2021.01.06.425648)>.

**License** AGPL-3

**URL** <https://dowser.readthedocs.io>

**BugReports** <https://github.com/immcantation/dowser/issues>

**LazyData** true

**BuildVignettes** true

**VignetteBuilder** knitr

**Encoding** UTF-8

**Depends** R (>= 4.4), ggplot2 (>= 3.4.0)

**Imports** airr, alakazam (>= 1.1.0), ape (>= 5.6), dplyr (>= 1.0),  
ggtree, graphics, gridExtra, markdown, methods, phangorn (>=  
2.7.1), phylotate, RColorBrewer, rlang, shazam (>= 1.1.1),  
stats, stringr, tidyselect, tidyr, utils, pwalgn, treeio

**Suggests** knitr, rmarkdown, testthat, BiocManager

**RoxygenNote** 7.3.3

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

**NeedsCompilation** no

**Author** Kenneth Hoehn [aut, cre],  
 Cole Jensen [aut],  
 Jessie Fielding [aut],  
 Hunter Melton [aut],  
 Susanna Marquez [ctb],  
 Jason Vander Heiden [ctb],  
 Steven Kleinstejn [aut, cph]

**Maintainer** Kenneth Hoehn <kenneth.b.hoehn@dartmouth.edu>

**Repository** CRAN

**Date/Publication** 2026-03-26 19:50:02 UTC

## Contents

airrClone-class . . . . .	4
BiopsyTrees . . . . .	5
bootstrapTrees . . . . .	5
buildBeast . . . . .	7
buildClonalGermline . . . . .	9
buildGermline . . . . .	10
buildIgphym1 . . . . .	12
buildPhylo . . . . .	13
buildPML . . . . .	14
buildPratchet . . . . .	16
buildRAXML . . . . .	17
calcRF . . . . .	18
checkDivergence . . . . .	19
collapseNodes . . . . .	20
colorTrees . . . . .	20
condenseTrees . . . . .	21
correlationTest . . . . .	22
createGermlines . . . . .	23
create_alignment . . . . .	26
create_height_prior . . . . .	26
create_max_height_prior . . . . .	27
create_MRCA_prior_germline . . . . .	27
create_MRCA_prior_observed . . . . .	28
create_root_freqs . . . . .	28
create_starting_tree . . . . .	29
create_traitset . . . . .	29
dfToFasta . . . . .	30
downsampleClone . . . . .	31
dowser . . . . .	32

ExampleAirr . . . . .	32
ExampleAirrTyCHE . . . . .	33
ExampleClones . . . . .	34
ExampleDbChangeo . . . . .	34
ExampleMixedClones . . . . .	35
ExampleMixedDb . . . . .	36
exportTrees . . . . .	38
findSwitches . . . . .	38
formatClones . . . . .	40
getAllSeqs . . . . .	43
getBootstraps . . . . .	44
getDivergence . . . . .	45
getGermline . . . . .	46
getNodeSeq . . . . .	47
getPalette . . . . .	47
getSeq . . . . .	48
getSkylines . . . . .	49
getSubclones . . . . .	50
getSubTaxa . . . . .	51
getTimeTrees . . . . .	51
getTimeTreesIterate . . . . .	53
getTrees . . . . .	54
IsotypeTrees . . . . .	56
makeAirrClone . . . . .	57
makeModelFile . . . . .	59
makeSkyline . . . . .	60
maskCodons . . . . .	61
maskSequences . . . . .	62
plotSkylines . . . . .	64
plotTrees . . . . .	64
readBEAST . . . . .	66
readFasta . . . . .	67
readIMGT . . . . .	68
readLineages . . . . .	68
readModelFile . . . . .	69
reconIgPhyML . . . . .	70
rerootTree . . . . .	71
resolveLightChains . . . . .	72
resolvePolytomies . . . . .	73
runCorrelationTest . . . . .	74
sampleCloneMultiGroup . . . . .	75
sampleClones . . . . .	76
scaleBranches . . . . .	76
stitchRegions . . . . .	77
stitchVDJ . . . . .	78
stopCodonCheck . . . . .	79
testPS . . . . .	80
testSC . . . . .	81

testSP . . . . .	82
TimeTrees . . . . .	84
treesToPDF . . . . .	85
writeCloneSequences . . . . .	86
writeLineageFile . . . . .	86
write_clones_to_xmls . . . . .	87
write_clone_to_xml . . . . .	89

<b>Index</b>	<b>91</b>
--------------	-----------

---

airrClone-class	<i>S4 class defining a clone in Dowser</i>
-----------------	--

---

## Description

airrClone defines a common data structure for perform lineage reconstruction 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

hgermline 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.

---

BiopsyTrees

*Example Ig lineage trees with biopsy reconstructions.*

---

### Description

Same as ExampleClones but with biopsies predicted at internal nodes

### Usage

BiopsyTrees

### 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

[BiopsyTrees](#)

---

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,  
  igphym1 = 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 <a href="#">formatClones</a>
bootstraps	number of bootstrap replicates to perform
nproc	number of cores to parallelize computations
trait	trait to use for parsimony models (required if igphym1 specified)
dir	directory where temporary files will be placed (required if igphym1 or dnapars specified)
id	unique identifier for this analysis (required if igphym1 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
igphym1	location of igphym1 executable if trait models desired
fixtrees	keep tree topologies fixed? (bootstrapping will not be performed)
quiet	amount of rubbish to print to console
rm_temp	remove temporary files (default=TRUE)
palette	deprecated
resolve	how should polytomies be resolved? 0=none, 1=max parsimony, 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 igphym1 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.

---

buildBeast	<i>Read in a directory from a BEAST run. Runs treeannotator and log-analyser.</i>
------------	---

---

**Description**

Read in a directory from a BEAST run. Runs treeannotator and loganalyser.

**Usage**

```
buildBeast(  
  data,  
  beast,  
  time,  
  template,  
  dir,  
  id,  
  mcmc_length = 1e+06,  
  resume_clones = NULL,  
  trait = NULL,  
  asr = FALSE,  
  full_posterior = FALSE,  
  log_every = "auto",  
  include_germline = TRUE,  
  nproc = 1,  
  quiet = 0,  
  burnin = 10,  
  low_ram = TRUE,  
  germline_range = c(-10000, 10000),  
  java = TRUE,  
  seed = NULL,  
  log_target = 10000,  
  trees = NULL,  
  tree_states = FALSE,  
  start_edge_length = 100,  
  start_date = NULL,  
  max_start_date = NULL,  
  germline_trait_value = "?",  
  ...  
)
```

**Arguments**

data	a list of airrClone objects
beast	location of beast binary directory (beast/bin)
time	Name of sample time column
template	XML template
dir	directory where temporary files will be placed.
id	unique identifier for this analysis
mcmc_length	Number of MCMC steps
resume_clones	Clones to resume for mcmc_length more steps
trait	Trait column used
asr	Log ancestral sequences?
full_posterior	Read un full distribution of parameters and trees?
log_every	Frequency of states logged. auto will divide mcmc_length by log_target
include_germline	Include germline in analysis?
nproc	Number of cores for parallelization. Uses at most 1 core per tree.
quiet	Amount of rubbish to print to console
burnin	Burnin percent (default 10)
low_ram	run with less memory (slightly slower)
germline_range	Possible date range of germline tip
java	Use the -java flag for BEAST run
seed	Use specified seed for the -seed option for BEAST
log_target	Target number of samples over mcmc_length
trees	optional list of starting trees, either phylo objects or newick strings
tree_states	Use states vector for starting tree
start_edge_length	edge length to use for all branches in starting tree
start_date	Starting date of time tree if desired
max_start_date	Maximum starting date of time tree if desired
germline_trait_value	trait value for germline, default '?' for ambiguous
...	Additional arguments for XML writing functions

**Value**

The input clones tibble with an additional column for the bootstrap replicate trees.

**See Also**

[getTimeTrees](#)

---

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,
  chain = "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 <a href="#">readIMGT</a>
chain	chain 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 <a href="#">buildGermline</a>

**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 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 = NULL,
  optimize = "lr",
  motifs = "FCH",
  hotness = "e,e,e,e,e,e",
  rates = NULL,
  asrc = 0.95,
  splitfreqs = FALSE,
  asrp = FALSE,
  trunkl = NULL,
  make_gyrep = TRUE,
  ...
)

```

### Arguments

clone	list of airrClone objects
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)
rates	comma delimited list showing which omega-defined partitions get a separate rate (e.g. omega=e,e rates=0,1).
asrc	Intermediate sequence cutoff probability
splitfreqs	Calculate codon frequencies on each partition separately?
asrp	Run ASRp?
trunkl	Set trunk length to specified number
make_gyrep	Create the grep file?
...	Additional arguments (not currently used)

### Details

Partition options in rate order:

- 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 heavy FWR, 1 for all CDRs, and 1 for light FWRs
- hlc: 3 omegas, 1 for all FWRs, 1 for heavy CDRs, and 1 for light CDRs
- hlcf: 4 omegas, 1 for each heavy FWR, 1 for heavy CDR, 1 for light FWR, and 1 for light CDR

### Value

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

---

buildPhylo

*Wrapper for alakazam::buildPhyloLineage*

---

### Description

Wrapper for alakazam::buildPhyloLineage

**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	sequence 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	<i>Wrapper for phangorn::optim.pml</i>
----------	--

---

**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,  
    optEdge = TRUE,  
    verbose = FALSE,  
    resolve_random = TRUE,  
    quiet = 0,  
    rep = NULL,  
    dir = NULL,  
    id = NULL,  
    asrp = FALSE  
  )
```

### Arguments

clone	airrClone object
seq	sequence 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
optEdge	Optimize edge lengths
verbose	Print error messages as they happen?
resolve_random	randomly resolve polytomies?
quiet	amount of rubbish to print to console
rep	current bootstrap replicate (experimental)
dir	A directory to save the codon table
id	The identifier value
asrp	Get the codon table?

### Value

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

---

buildPratchet	<i>Wrapper for phangorn::pratchet</i>
---------------	---------------------------------------

---

### 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	sequence 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.

---

 buildRAxML

*Wrapper to build RAxML-ng trees and infer intermediate nodes*


---

## Description

Wrapper to build RAxML-ng trees and infer intermediate nodes

## Usage

```

buildRAxML(
  clone,
  exec,
  seq = "sequence",
  sub_model = "GTR",
  partition = NULL,
  rseed = 28,
  name = "run",
  starting_tree = NULL,
  data_type = "DNA",
  from_getTrees = FALSE,
  rm_files = TRUE,
  asr = TRUE,
  rep = 1,
  dir = NULL,
  n_starts = NULL,
  ...
)

```

## Arguments

clone	list of airrClone objects
exec	RAxML-ng executable
seq	the phylo_seq option does this clone uses. Possible options are "sequence", "hlsequence", or "lsequence"
sub_model	The DNA model to be used. GTR is the default.
partition	A parameter that determines how branches are reported when partitioning. Options include NULL (default), scaled, unlinked, and linked
rseed	The random seed used for the parsimony inferences. This allows you to reproduce your results.
name	specifies the name of the output file
starting_tree	specifies a user starting tree file name and path in Newick format
data_type	Specifies what format your data is in, DNA or AA
from_getTrees	A logical that indicates if the desired starting tree is from getTrees and not a newick file

rm_files	remove temporary files?
asr	computes the marginal ancestral states of a tree
rep	Which repetition of the tree building is currently being run. Mainly for getBoostraps.
dir	Where the output files are to be made.
n_starts	Number of max parsimony starting trees (default is 10 pars + 10 random)
...	Additional arguments (not currently used)

**Value**

phylo object created by RAxML-ng with nodes attribute containing reconstructed sequences.

---

calcRF

*Finds the Robinson-Fould's cluster distance between phylogenies.*

---

**Description**

calcRF Calculates the RF distance between two phylogenetic trees with the same tips and tip labels.

**Usage**

```
calcRF(tree_1, tree_2, nproc = 1)
```

**Arguments**

tree_1	A phylo object
tree_2	A phylo object
nproc	Number of cores to use for calculations.

**Value**

The RF cluster value for the two input trees.

---

checkDivergence	<i>Compare divergence along a tree in terms of mutations (sum of branches) for each tip and reconstructed internal node to its Hamming distance from the germline. Divergence should never be less than Hamming distance. A threshold of -1 is used to represent 1 full mutation difference. The function will throw a warning if any trees cross this threshold</i>
-----------------	--

---

### Description

Compare divergence along a tree in terms of mutations (sum of branches) for each tip and reconstructed internal node to its Hamming distance from the germline. Divergence should never be less than Hamming distance. A threshold of -1 is used to represent 1 full mutation difference. The function will throw a warning if any trees cross this threshold

### Usage

```
checkDivergence(  
  clones,  
  threshold = -1,  
  verbose = TRUE,  
  germline = "Germline",  
  data_type = "DNA"  
)
```

### Arguments

clones	a tibble of clones and trees, output from <a href="#">getTrees</a>
threshold	Minimum allowed value of divergence minus Hamming distance
verbose	Print whether all trees passed
germline	ID of the tree's predicted germline sequence
data_type	The type of data being used. Either "DNA" (default) or "AA"

### Value

tibble showing the clone\_id, sequence\_id, as well as tree-based divergence, hamming distance, and difference between the two.

---

collapseNodes	<i>Collapse internal nodes with the same predicted sequence</i>
---------------	---

---

**Description**

collapseNodes Node collapsing function.

**Usage**

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

**Arguments**

trees	a tibble of airrClone objects, the output of <a href="#">getTrees</a>
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	<i>Get a color palette for a predefined set of trait values</i>
------------	---

---

**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 <a href="#">getPalette</a> )
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	<i>Condense a set of equally parsimonious node labels into a single tree</i>
---------------	--

---

**Description**

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

**Usage**

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

**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	<i>createGermlines</i> Determine consensus clone sequence and create germline for clone
-----------------	---

---

**Description**

[createGermlines](#) Determine consensus clone sequence and create germline for clone

**Usage**

```
createGermlines(
  data,
  references,
  locus = "locus",
  trim_lengths = FALSE,
  force_trim = FALSE,
  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 <a href="#">readIMGT</a>
locus	Name of the locus column in the input data
trim_lengths	Remove trailing Ns from seq column if length different from germline?
force_trim	Remove all characters from sequence if different from germline? (not recommended)
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 <a href="#">buildGermline</a>

## 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)
```

---

create_alignment	<i>Takes an airr clone object and returns BEAST2 Alignment xml of the sequences</i>
------------------	---

---

**Description**

Takes an airr clone object and returns BEAST2 Alignment xml of the sequences

**Usage**

```
create_alignment(clone, id, include_germline_as_tip)
```

**Arguments**

clone	an airrClone object
id	unique identifier for this analysis
include_germline_as_tip	include the germline as a tip in the alignment?

**Value**

String of BEAST2 Alignment and TaxonSet xml

---

create_height_prior	<i>Takes an airr clone object and returns BEAST2 XML to set a height prior</i>
---------------------	--

---

**Description**

Takes an airr clone object and returns BEAST2 XML to set a height prior

**Usage**

```
create_height_prior(clone, id, start_date)
```

**Arguments**

clone	an airrClone object
id	unique identifier for this analysis
start_date	starting date to use as prior, in forward time

**Value**

String of XML setting the height prior

---

`create_max_height_prior`

*Takes an airr clone object and returns BEAST2 XML to set a maximum height prior*

---

**Description**

Takes an airr clone object and returns BEAST2 XML to set a maximum height prior

**Usage**

```
create_max_height_prior(clone, id, max_start_date)
```

**Arguments**

`clone`            an airrClone object  
`id`                unique identifier for this analysis  
`max_start_date` max start date to use for prior, in forward time

**Value**

String of XML setting the MRCA prior of the observed sequences

---

`create_MRCA_prior_germline`

*Takes an airr clone object and returns BEAST2 XML for MRCA prior of the germline sequence*

---

**Description**

Takes an airr clone object and returns BEAST2 XML for MRCA prior of the germline sequence

**Usage**

```
create_MRCA_prior_germline(clone, id, germline_range)
```

**Arguments**

`clone`            an airrClone object  
`id`                unique identifier for this analysis  
`germline_range` Possible date range of germline tip

**Value**

String of XML setting the MRCA prior of the germline sequence

create\_MRCA\_prior\_observed

*Takes an airr clone object and returns BEAST2 XML for MRCA prior of the observed sequences*

---

### **Description**

Takes an airr clone object and returns BEAST2 XML for MRCA prior of the observed sequences

### **Usage**

```
create_MRCA_prior_observed(clone, id)
```

### **Arguments**

clone	an airrClone object
id	unique identifier for this analysis

### **Value**

String of XML setting the MRCA prior of the observed sequences

---

create\_root\_freqs

*Takes an airr clone object and returns BEAST2 rootfreqs xml of the germline*

---

### **Description**

Takes an airr clone object and returns BEAST2 rootfreqs xml of the germline

### **Usage**

```
create_root_freqs(clone, id)
```

### **Arguments**

clone	an airrClone object
id	unique identifier for this analysis

### **Value**

String of XML setting the root frequencies to the germline sequence

---

create\_starting\_tree *Takes an airr clone object and tree and returns BEAST2 XML for setting the starting tree*

---

### Description

Takes an airr clone object and tree and returns BEAST2 XML for setting the starting tree

### Usage

```
create_starting_tree(
  clone,
  id,
  tree,
  include_germline_as_tip,
  tree_states,
  start_edge_length
)
```

### Arguments

clone	an airrClone object
id	unique identifier for this analysis
tree	starting tree, either a phylo object or a newick string
include_germline_as_tip	include the germline as a tip
tree_states	use states in the starting tree?
start_edge_length	edge length to use for all branches in starting tree

### Value

String of XML setting the starting tree

---

create\_traitset *Takes an airr clone object and returns BEAST2 XML for a trait/traitSet from a column*

---

### Description

Takes an airr clone object and returns BEAST2 XML for a trait/traitSet from a column

**Usage**

```

create_traitset(
  clone,
  trait_name,
  column,
  id,
  trait_data_type = NULL,
  isSet = FALSE,
  include_germline_as_tip = FALSE,
  germline_value = "?"
)

```

**Arguments**

clone	an airrClone object
trait_name	name of the trait
column	column in the clone data to use for the trait
id	unique identifier for this analysis
trait_data_type	optional data type for the trait
isSet	is this a traitSet (TRUE) or a trait (FALSE)?
include_germline_as_tip	include the germline as a tip
germline_value	trait value for germline, default '?' for ambiguous

**Value**

String of XML of the trait or traitSet

---

dfToFasta

---

*Write a fasta file of sequences readFasta reads a fasta file*


---

**Description**

Write a fasta file of sequences readFasta reads a fasta file

**Usage**

```

dfToFasta(
  df,
  file,
  id = "sequence_id",
  seq = "sequence",
  imgt_gaps = FALSE,
  columns = NULL
)

```

**Arguments**

df	dataframe of sequences
file	FASTA file for output
id	Column name of sequence ids
seq	Column name of sequences
imgt_gaps	Keep IMGT gaps if present?
columns	vector of column names to append to sequence id

**Value**

File of FASTA formatted sequences

---

downsampleClone	downsampleClone <i>Down-sample clone to maximum tip/switch ratio</i>
-----------------	--

---

**Description**

downsampleClone Down-sample clone to maximum tip/switch ratio

**Usage**

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

**Arguments**

clone	an <a href="#">airrClone</a> object
trait	trait considered for rarefaction <a href="#">getTrees</a>
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	<i>The dowser package</i>
--------	---------------------------

---

### 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, Kleinstei SH (2022) Phylogenetic analysis of migration, differentiation, and class switching in B cells. PLoS Computational Biology. <https://doi.org/10.1371/journal.pcbi.1009885>

---

ExampleAirr	<i>Example AIRR database</i>
-------------	------------------------------

---

### 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](#)

---

ExampleAirrTyCHE

*Example AIRR database for TyCHE*

---

**Description**

A small example database from simple.

**Usage**

ExampleAirrTyCHE

**Format**

A data.frame with the following AIRR style columns:

- `sequence_id`: Sequence identifier
- `sequence_alignment`: IMGT-gapped observed sequence.
- `germline_alignment`: IMGT-gapped germline sequence.
- `v_call`: V region allele assignments.
- `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_time`: Time point of the sample.
- `location`: Location of tissue from which the sample was taken.
- `clone_id`: Clonal group identifier.

**References**

1. Pre-submission.

---

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](#)

---

ExampleMixedClones      *Example Multiple Partition Trees*

---

## Description

A small example database subset from Turner, J. S. et al. Human germinal centres engage memory and naive B cells after influenza vaccination. Nature 586, 127–132 (2020).

## Usage

ExampleMixedClones

## Format

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

- clone\_id: Clonal cluster
- data: List of airrClone objects
- locus: Locus identifier.
- seqs: Number of sequences
- igphyml\_partitioned\_trees: IgPhyML partitioned tree
- raxml\_partitioned\_trees: RAxML partitioned tree

---

ExampleMixedDb

*Example Change-O database*

---

### Description

A small example database subset from Turner, J. S. et al. Human germinal centres engage memory and naive B cells after influenza vaccination. *Nature* 586, 127–132 (2020).

### Usage

ExampleMixedDb

### Format

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

- `sequence_id`: Sequence identifier
- `sequence`: B cell sequence
- `productive`: A logical indicating if the sequence is productive.
- `v_call`: V region allele assignments.
- `d_call`: D region allele assignments.
- `j_call`: J region allele assignments.
- `sequence_alignment`: Sequence alignment.
- `germline_alignment`: Germline alignment without gaps.
- `junction`: Junction
- `juncation_aa`: Junction aa
- `vj_inframe`: A logical to see if the vj genes are in frame
- `stop_codon`: A indicator if there is a stop codon within the alignment
- `locus`: Locus identifier.
- `v_sequence_start`: Where the V gene starts
- `v_sequence_end`: Where the V gene ends
- `v_germline_start`: Where the V germline starts
- `v_germline_end`: Where the V germline ends
- `np1_length`: Length of np1
- `d_sequence_start`: Where the D gene starts
- `d_sequence_end`: Where the D gene ends
- `d_germline_start`: Where the D germline starts
- `d_germline_end`: Where the D germline ends
- `np2_length`: Length of np2
- `j_sequence_start`: Where the J gene starts

- `j_sequence_end`: Where the J gene ends
- `j_germline_start`: Where the J germline starts
- `j_germline_end`: Where the J germline ends
- `junction_length`: Length of the junction region in nucleotides.
- `v_score`: V score
- `v_identity`: Identity score of V
- `v_support`: V support
- `d_score`: D score
- `d_identity`: D identity
- `d_support`: D support
- `j_score`: J score
- `j_support`: J support
- `j_identity`: J identity
- `cell_id`: Cell identifier
- `consensus_count`: Consensus count
- `indels`: Logical if indels are present
- `sequence_vdj`: VDJ sequence
- `v_germ_start_vdj`: Where the V germline starts on the VDJ
- `v_germ_end_vdj`: Where the V germline ends on the VDJ
- `subject`: Subject identifier
- `timepoint`: Day the sample was taken
- `cell_type`: Type of cell
- `replicate`: Replicate number
- `clone_id`: Change-O assignment clonal group identifier.
- `seq_type`: Identifier of data type (10x)
- `vj_gene`: VJ gene
- `vj_alt_gene`: Alternative VJ gene
- `v_germline_length`: Length of the V germline segment
- `d_germline_length`: Length of the D germline segment
- `j_germline_length`: Length of the J germline segment
- `germline_alignment_d_mask`: Germline alignment with gaps

---

exportTrees	<i>Exports the phylogenetic trees from the airrClone object</i>
-------------	---

---

**Description**

exportTrees Exports phylogenetic trees

**Usage**

```
exportTrees(clones, filepath, tree_column = "trees", ...)
```

**Arguments**

clones	tibble airrClone objects, the output of <a href="#">formatClones</a>
filepath	The file path for where the trees will be saved
tree_column	The name of the column that contains the trees
...	additional arguments to be passed

---

findSwitches	<i>Create a bootstrap distribution for clone sequence alignments, and estimate trees for each bootstrap replicate.</i>
--------------	--

---

**Description**

findSwitches Phylogenetic bootstrap function.

**Usage**

```
findSwitches(
  clones,
  permutations,
  trait,
  igphym1,
  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,
    keptrees = FALSE,
    lfile = NULL,
    seq = NULL,
    boot_part = "locus",
    force_resolve = FALSE,
    ...
)

```

### Arguments

clones	tibble airrClone objects, the output of <a href="#">formatClones</a>
permutations	number of bootstrap replicates to perform
trait	trait to use for parsimony models
igphym1	location of igphym1 executable
fixtrees	keep tree topologies fixed? (bootstrapping will not be performed)
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 igphym1 or dnapars specified)
id	unique identifier for this analysis (required if igphym1 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	deprecated
resolve	how should polytomies be resolved? 0=none, 1=max parsimony, 2=max ambiguity + polytomy skipping, 3=max ambiguity
rep	current bootstrap replicate (experimental)
keptrees	keep trees estimated from bootstrap replicates? (TRUE)
lfile	lineage file input to igphym1 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.

Trait values cannot contain values N, UCA, or NTIP. These are reserved for use by test statistic functions.

## 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 duplicate sequences, and merges annotations associated with duplicate sequences. If specified, it will un-merge duplicate sequences with different values specified in the `traits` 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",
  subgroup = "clone_subgroup",
  id = "sequence_id",
  germ = "germline_alignment_d_mask",
  v_call = "v_call",
  j_call = "j_call",
  junc_len = "junction_length",
  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,
  cell = "cell_id",
  locus = "locus",
  traits = NULL,
  mod3 = TRUE,
  randomize = TRUE,
  use_regions = TRUE,
  dup_singles = FALSE,
  nproc = 1,
  chain = "H",
  heavy = "IGH",
  filterstop = TRUE,
  minseq = 2,
  split_light = FALSE,
  light_traits = FALSE,
  majoronly = FALSE,
  columns = NULL
)

```

**Arguments**

data	data.frame containing the AIRR or Change-O data for a clone. See <a href="#">makeAirrClone</a> for required columns and their defaults
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.
subgroup	name of the column containing the identifier for the subgroup.

id	name of the column containing sequence identifiers.
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.
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?
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 multiple 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?
nproc	number of cores to parallelize formatting over.
chain	if HL, include light chain information if available.
heavy	name of heavy chain locus (default = "IGH")
filterstop	only use sequences that do not contain an in-frame stop codon
minseq	minimum number of sequences per clone
split_light	split or lump subgroups? See resolveLightChains.
light_traits	Include the traits from the light chain when concatenating and collapsing trees?
majoronly	only return largest subgroup and sequences without light chains
columns	additional data columns to include in output

**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,], traits="sample_id")
```

---

getAllSeqs

*Return all tip and internal node sequences*

---

**Description**

getNodeSeq Sequence retrieval function.

**Usage**

```
getAllSeqs(data, imgt_gaps = TRUE)
```

**Arguments**

`data` a tibble of [airrClone](#) objects with reconstructed trees, the output of [getTrees](#)  
`imgt_gaps` include a column of gapped sequences?

**Details**

Column names: clone\_id = clone id node\_id = name of node, either the sequence name if a tip or Node<number> if internal node node = node number in tree. Tips are nodes 1:<number of tips>. locus = locus of sequence sequence = ungapped sequence, either observed for tips or reconstructed for internal nodes sequence\_alignment = sequence with IMGT gaps (optional)

**Value**

A tibble with sequence information for each tip and internal node of a set of trees.

**See Also**

[getTrees](#) [getNodeSeq](#)

---

getBootstraps	<i>Creates a bootstrap distribution for clone sequence alignments, and returns estimated trees for each bootstrap replicate as a nested list as a new input tibble column.</i>
---------------	--

---

### Description

getBootstraps Phylogenetic bootstrap function.

### Usage

```
getBootstraps(
  clones,
  bootstraps,
  nproc = 1,
  bootstrap_nodes = TRUE,
  dir = NULL,
  id = NULL,
  build = "pratchet",
  exec = NULL,
  quiet = 0,
  rm_temp = TRUE,
  rep = NULL,
  seq = NULL,
  boot_part = "locus",
  by_codon = TRUE,
  starting_tree = FALSE,
  switches = FALSE,
  ...
)
```

### Arguments

clones	tibble airrClone objects, the output of <a href="#">formatClones</a>
bootstraps	number of bootstrap replicates to perform
nproc	number of cores to parallelize computations
bootstrap_nodes	a logical if the the nodes for each tree in the trees column (required) should report their bootstrap value
dir	directory where temporary files will be placed (required if igphym1 or dnapars specified)
id	unique identifier for this analysis (required if igphym1 or dnapars specified)
build	program to use for tree building (phangorn, dnapars, igphym1)
exec	location of desired phylogenetic executable
quiet	amount of rubbish to print to console

rm_temp	remove temporary files (default=TRUE)
rep	current bootstrap replicate (experimental)
seq	column name containing sequence information
boot_part	is "locus" bootstrap columns for each locus separately
by_codon	a logical if the user wants to bootstrap by codon or by nucleotide. Default (codon based bootstrapping) is TRUE.
starting_tree	An indicator to use the existing trees column as the starting trees for RAxML
switches	a logical indicator to allow findSwitches to do permutations.
...	additional arguments to be passed to tree building program

**Value**

The input clones tibble with an additional column for the bootstrap replicate trees.

---

getDivergence	<i>Get divergence from root of tree for each tip</i>
---------------	--

---

**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	<i>getGermline</i> 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 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	<i>Return IMGT gapped sequence of specified tree node</i>
------------	---

---

**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 <a href="#">getTrees</a>
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	<i>Get a color palette for a predefined set of trait values. 'Germline' defaults to black unless specified.</i>
------------	---

---

**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 <a href="#">getTrees</a>
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](#)

---

`getSkylines`*Make data frames for Bayesian skyline plots*

---

**Description**`makeSkylines`**Usage**

```
getSkylines(  
  clones,  
  dir,  
  id,  
  time,  
  burnin = 10,  
  bins = 100,  
  verbose = 0,  
  forward = TRUE,  
  nproc = 1,  
  max_height = c("min", "median", "mean", "max")  
)
```

**Arguments**

<code>clones</code>	clone tibble
<code>dir</code>	directory of BEAST trees file
<code>id</code>	unique identifier for this analysis
<code>time</code>	name of time column
<code>burnin</code>	Burnin percent (default 10)
<code>bins</code>	number of bins for plotting
<code>verbose</code>	if 1, print name of clones
<code>forward</code>	plot in forward or (FALSE) backward time?
<code>nproc</code>	processors for parallelization (by clone)
<code>max_height</code>	max height to use (min, median, mean, max)

**Details**

Burnin set from `readBEAST` or `getTrees`

**Value**

Bayesian Skyline values for given clone

---

```
getSubclones          #' Deprecated! Use resolveLightChains
```

---

### 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

**Value**

a tibble containing

---

getSubTaxa	<i>Get the tip labels as part of a clade defined by an internal node</i>
------------	--

---

**Description**

getSubTaxa Gets the tip labels from a clade

**Usage**

```
getSubTaxa(node, tree)
```

**Arguments**

node	node number that defines the target clade
tree	phylo object

**Value**

A vector containing tip labels of the clade

**Examples**

```
# Get taxa from all subtrees
data(BiopsyTrees)
tree <- BiopsyTrees$trees[[8]]
all_subtrees <- lapply(1:length(tree$nodes), function(x) getSubTaxa(x, tree))
```

---

getTimeTrees	<i>Estimate time trees by running BEAST on each clone Applies XML template to each clone</i>
--------------	--

---

**Description**

getTimeTrees Tree building function.

**Usage**

```

getTimeTrees(
  clones,
  template,
  beast,
  dir,
  id,
  time,
  mcmc_length = 3e+07,
  log_every = "auto",
  burnin = 10,
  trait = NULL,
  resume_clones = NULL,
  nproc = 1,
  quiet = 0,
  rm_temp = FALSE,
  include_germline = TRUE,
  seq = "sequence",
  germline_range = c(-10000, 10000),
  java = TRUE,
  seed = NULL,
  log_target = 10000,
  tree_states = FALSE,
  trees = NULL,
  germline_trait_value = "?",
  ...
)

```

**Arguments**

clones	a tibble of <code>airrClone</code> objects, the output of <a href="#">formatClones</a>
template	XML template
beast	location of beast binary directory (beast/bin)
dir	directory where temporary files will be placed.
id	unique identifier for this analysis
time	Name of sample time column
mcmc_length	Number of MCMC steps
log_every	Frequency of states logged. "auto" will divide mcmc_length by log_target
burnin	Burnin percent (default 10)
trait	Trait column to be used
resume_clones	Clones to resume for mcmc_length more steps
nproc	Number of cores for parallelization. At most 1 core/tree can be used.
quiet	amount of rubbish to print to console
rm_temp	remove temporary files (default=TRUE)

include_germline	Include germline sequence in analysis?
seq	Sequence column in data
germline_range	Possible date range of germline tip
java	Use the -java flag for BEAST run
seed	Use specified seed for the -seed option for BEAST
log_target	Target number of samples from MCMC chain
tree_states	Use states vector for starting tree
trees	optional list of starting trees, either phylo objects or newick strings
germline_trait_value	trait value for germline, default '?' for ambiguous
...	Additional arguments passed to tree building programs

**Details**

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

**Value**

A tibble with a column of phylo objects and parameters column

**See Also**

[getTrees](#), [readBEAST](#)

---

getTimeTreesIterate	<i>Iteratively resume getTimeTrees until convergence, as defined by all parameters (except those in ignore vector) having ESS greater than or equal to the specified ess_cutoff</i>
---------------------	---

---

**Description**

getTimeTreesIterate Iteratively resume getTimeTrees til convergence.

**Usage**

```
getTimeTreesIterate(
  clones,
  iterations = 10,
  ess_cutoff = 200,
  ignore = c("traitfrequencies"),
  quiet = 0,
  ...
)
```

**Arguments**

clones	a tibble of airrClone objects, the output of <a href="#">formatClones</a>
iterations	Maximum number of times to resume MCMC chain
ess_cutoff	Minimum number of ESS for all parameters
ignore	Vector of parameters to ignore for ESS calculation
quiet	quiet notifications if > 0
...	Additional arguments for getTimeTrees

**Details**

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

**Value**

A tibble of tidytree and airrClone objects.

---

getTrees	<i>Estimate lineage tree topologies, branch lengths, and internal node states if desired</i>
----------	--

---

**Description**

getTrees Tree building function.

**Usage**

```
getTrees(
  clones,
  trait = NULL,
  id = NULL,
  dir = NULL,
  modelfile = NULL,
  build = "pratchet",
  exec = NULL,
  igphym1 = NULL,
  fixtrees = FALSE,
  nproc = 1,
  quiet = 0,
  rm_temp = TRUE,
  palette = NULL,
  seq = NULL,
  collapse = FALSE,
  check_divergence = TRUE,
  ...
)
```

**Arguments**

<code>clones</code>	a tibble of <code>airrClone</code> objects, the output of <a href="#">formatClones</a>
<code>trait</code>	trait to use for parsimony models (required if <code>igphyml</code> specified)
<code>id</code>	unique identifier for this analysis (required if <code>igphyml</code> or <code>dnapars</code> specified)
<code>dir</code>	directory where temporary files will be placed.
<code>modelfile</code>	file specifying parsimony model to use
<code>build</code>	program to use for tree building ( <code>pratchet</code> , <code>pml</code> , <code>dnapars</code> , <code>dnaml</code> , <code>igphyml</code> , <code>raxml</code> )
<code>exec</code>	location of desired phylogenetic executable
<code>igphyml</code>	optional location of <code>igphyml</code> executable for parsimony
<code>fixtrees</code>	if TRUE, use supplied tree topologies
<code>nproc</code>	number of cores to parallelize computations
<code>quiet</code>	amount of rubbish to print to console
<code>rm_temp</code>	remove temporary files (default=TRUE)
<code>palette</code>	deprecated
<code>seq</code>	column name containing sequence information
<code>collapse</code>	Collapse internal nodes with identical sequences?
<code>check_divergence</code>	run <a href="#">checkDivergence</a> on trees?
<code>...</code>	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 `phangorn::pratchet` to estimate maximum parsimony tree topologies, and `ape::acctrans` to estimate branch lengths. If `igphyml` 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 be 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](#), [buildRAXML](#)

## Examples

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

## Not run:
data(ExampleClones)

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

## End(Not run)
```

---

IsotypeTrees

*Example Ig lineage trees with isotype reconstructions.*

---

## Description

Same as ExampleClones but with isotypes predicted at internal nodes

## Usage

IsotypeTrees

## 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

[IsotypeTrees](#)

---

makeAirrClone	<i>Generate a airrClone object for lineage construction</i>
---------------	---

---

### Description

makeAirrClone takes a data.frame with AIRR or Change-O style columns as input and masks gap positions, masks ragged ends, removes duplicate 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",  
  subgroup = "clone_subgroup",  
  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,  
  light_traits = 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.
subgroup	name of the column containing the identifier for the subgroup.
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 multiple 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?
light_traits	Include the traits from the light chain when concatenating and collapsing trees?

## Details

The input data.frame (data) must contain 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 be 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 building 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, exceptions = NULL)
```

**Arguments**

file	model file name to write.
states	vector of states to include in model.
constraints	constraints to add to model.
exceptions	vector of comma-separated states that are exceptions to constraints

**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](#)

---

makeSkyline	<i>get values for Bayesian Skyline plot</i>
-------------	---

---

**Description**

makeSkyline

**Usage**

```
makeSkyline(  
  logfile,  
  treesfile,  
  burnin,  
  bins = 100,  
  youngest = 0,  
  clone_id = NULL,  
  max_height = c("min", "median", "mean", "max")  
)
```

**Arguments**

logfile	Beast log file
treesfile	BEAST trees file
burnin	Burnin percentage (1-100)
bins	number of bins for plotting
youngest	timepoint of the most recently tip sampled (if 0, backward time used)
clone_id	name of the clone being analyzed (if desired)
max_height	max height to use (min, median, mean, max)

**Value**

Bayesian Skyline values for given clone

---

maskCodons	maskCodons <i>Masks codons split by insertions</i>
------------	--

---

**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 (pwalgn::pairwiseAlignment)
gap_extension	gap extension penalty (pwalgn::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](#), `pwalgn::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](#), [pwalgn::pairwiseAlignment](#).

---

plotSkylines	<i>Simple function for plotting Bayesian skyline plots</i>
--------------	--

---

**Description**

plotSkylines Simple Bayesian skyline plots

**Usage**

```
plotSkylines(clones, file = NULL, width = 8.5, height = 11, ...)
```

**Arguments**

clones	output from getTrees using BEAST
file	pdf file name for printing plots
width	width of plot in inches if file specified
height	height of plot in inches if file specified
...	optional arguments passed to grDevices::pdf

**Value**

if no file specified, a list of ggplot objects. If file specified will plot to specified file

**See Also**

[getSkylines](#) [readBEAST](#) [getTrees](#)

---

plotTrees	<i>Plot a tree with colored internal node labels using ggtree</i>
-----------	---

---

**Description**

plotTrees plots a tree or group of trees

**Usage**

```
plotTrees(  
  trees,  
  nodes = FALSE,  
  tips = NULL,  
  tipsize = NULL,  
  scale = 0.01,  
  palette = "Dark2",  
  base = FALSE,  
  show_occupancy = FALSE,
```

```

    layout = "rectangular",
    node_nums = FALSE,
    tip_nums = FALSE,
    title = TRUE,
    labelsize = NULL,
    common_scale = FALSE,
    ambig = "grey",
    bootstrap_scores = FALSE,
    tip_palette = NULL,
    node_palette = NULL,
    guide_title = NULL,
    branch_lengths = NULL,
    pch = 16
  )

```

### 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
palette	color palette for tips and/or nodes. Can supply a named vector for all tip states, or a palette named passed to ggplot2::scale_color_brewer (e.g. "Dark2", "Paired", "Set1") or ggplot2::scale_color_distiller (e.g. RdYIBu) or
base	recursion base case (don't edit)
show_occupancy	if plotting trees from an expectedOccupancy model in TyCHE, will color branch lengths by expected occupancy in the first state if true. Requires palette to be specified as a named vector in order of: c(state1=color1, state2+state1=color2, state2=color3).
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	stretch plots so branches are on same scale? determined by sequence with highest divergence
ambig	How to color ambiguous node reconstructions? (grey or blend)
bootstrap_scores	Show bootstrap scores for internal nodes? See getBootstraps.
tip_palette	deprecated, use palette
node_palette	deprecated, use palette
guide_title	Title of color guide. Defaults to tips variable if specified.
branch_lengths	Use branch lengths? Use "none" if not.
pch	Numeric tip/node shape. If >20 will use "fill" instead of "color"

### Details

Function uses `ggtree` functions to plot tree topologies estimated by `getTrees`, and `findSwitches`. Object can be further modified with `ggtree` functions. Please check out <https://bioconductor.org/packages/devel/bioc/vignettes/ggtree> 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]]
```

---

readBEAST

*Reads in a BEAST output directory*

---

### Description

readBEAST Reads in data from BEAST output directory

### Usage

```
readBEAST(
  clones,
  dir,
  id,
  beast,
  burnin = 10,
  trait = NULL,
  nproc = 1,
  quiet = 0,
  full_posterior = FALSE,
  asr = FALSE,
  low_ram = TRUE
)
```

**Arguments**

clones	either a tibble (getTrees) or list of airrClone objects
dir	directory where BEAST output files have been placed.
id	unique identifier for this analysis
beast	location of beast binary directory (beast/bin)
burnin	percent of initial tree samples to discard (default 10)
trait	Trait column used
nproc	Number of cores for parallelization. Uses at most 1 core per tree.
quiet	amount of rubbish to print to console
full_posterior	Read un full distribution of parameters and trees?
asr	Log ancestral sequences?
low_ram	run with less memory (slightly slower)

**Value**

If data is a tibble, then the input clones tibble with additional columns for trees and parameter estimates given the specified burnin. If input is just a list of airrClone objects, it will return the corresponding list of trees given the burnin

---

readFasta	<i>Read a fasta file into a list of sequences</i>	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	<i>readIMGT read in IMGT database</i>
----------	---------------------------------------

---

### 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](https://changeo.readthedocs.io/en/stable/examples/igblast.html) for example of how to download.

### Value

List of lists, leading to IMGT-gapped nucleotide sequences. Structure of object is list[[locus]][[segment]] locus refers to locus (e.g. IGH, IGK, TRA) segment refers to gene segment category (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 ExampleDb[1,]
vdj_dir <- system.file("extdata", "germlines", "imgt", "human", "vdj", package="dowser")
imgt <- readIMGT(vdj_dir)
```

---

readLineages	<i>Read in all trees from a lineages file</i>
--------------	---

---

### Description

Read in all trees from a lineages file

**Usage**

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

**Arguments**

file	IgPhyML lineage file
states	states in parsimony model
palette	deprecated
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	<i>Read in a parsimony model file</i>
---------------	---------------------------------------

---

**Description**

readModelFile Filler

**Usage**

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

**Arguments**

file	parsimony 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,
  igphym1 = "igphym1",
  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
igphym1	location of igphym1 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	deprecated
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	<i>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.</i>
------------	---

---

**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

---

resolveLightChains      *Define subgroups within clones based on light chain rearrangements*

---

### Description

resolveLightChains resolve light chain V and J subgroups within a clone

### Usage

```
resolveLightChains(
  data,
  nproc = 1,
  minseq = 1,
  locus = "locus",
  heavy = "IGH",
  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",
  pad_ends = TRUE
)
```

### Arguments

data	a tibble containing heavy and light chain sequences with clone_id
nproc	number of cores for parallelization
minseq	minimum number of sequences per clone
locus	name of column containing locus values
heavy	value of heavy chains in locus column. All other values will be treated as light chains
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
pad_ends	pad sequences within a clone to same length?

### Details

1. Make temporary array containing light chain clones 2. Enumerate all possible V, J, and junction length 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 1-6 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. Cells with missing light chains are grouped with their subgroup with the closest matching heavy chain (Hamming distance) then the largest and lowest index subgroup if ties are present.

Outputs of the function are 1. clone\_subgroup which identifies the light chain VJ rearrangement that sequence belongs to within it's clone 2. clone\_subgroup\_id which combines the clone\_id variable and the clone\_subgroup variable by a "\_". 3. vj\_cell which combines the vj\_gene and vj\_alt\_cell columns by a ",".

### Value

a tibble containing the same data as inputting, but with the column clone\_subgroup added. This column contains subgroups within clones that contain distinct light chain V and J genes, with at most one light chain per cell.

---

resolvePolytomies	<i>Resolve polytomies to have the minimum number of single timepoint clades</i>
-------------------	---

---

### 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 time-point 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](#).

---

sampleCloneMultiGroup sampleCloneMultiGroup *Down-sample clone to specified size with one or multiple groups to sample evenly*

---

**Description**

sampleCloneMultiGroup Down-sample clone to specified size with one or multiple groups to sample evenly

**Usage**

```
sampleCloneMultiGroup(clone, size, weight = NULL, group = NULL)
```

**Arguments**

clone	an <a href="#">airrClone</a> object
size	target size
weight	column for weighting sample probability
group	column(s) to sample evenly among group

**Value**

a down-sampled `airrClone` object

---

sampleClones	sampleClones <i>Down-sample clones to specified size</i>
--------------	--

---

**Description**

sampleClones Down-sample clones to specified size

**Usage**

```
sampleClones(clones, size, weight = NULL, group = NULL)
```

**Arguments**

clones	a tibble of <code>airrClone</code> objects
size	target size
weight	column for weighting sample probability
group	column (or columns) to sample evenly among groups

**Value**

The input object with sequences down-sampled

---

scaleBranches	<i>Scale branch lengths to represent either mutations or mutations per site.</i>
---------------	--

---

**Description**

scaleBranches Branch length scaling function.

**Usage**

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

**Arguments**

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

**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	<i>stitchRegions</i> Similar to <i>stitchVDJ</i> but with segment IDs instead of nucleotides
---------------	--

---

**Description**

[stitchRegions](#) Similar to [stitchVDJ](#) but with segment IDs instead of nucleotides

**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 <a href="#">getGermline</a>
d_seq	germline D segment sequence from <a href="#">getGermline</a>
j_seq	germline J segment sequence from <a href="#">getGermline</a>
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	<i>stitchVDJ combines germline gene segments to a single string</i>
-----------	---

---

**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 <a href="#">getGermline</a>
d_seq	germline D segment sequence from <a href="#">getGermline</a>
j_seq	germline J segment sequence from <a href="#">getGermline</a>

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.

---

stopCodonCheck	<i>Check whether sequences have in-frame premature stop codons (PTCs)</i>
----------------	---

---

**Description**

Check whether sequences have in-frame premature stop codons (PTCs)

**Usage**

```
stopCodonCheck(sequences, nproc = 1, check = TRUE)
```

**Arguments**

sequences	Vector of nucleotide sequences in desired reading frame
nproc	Number of cores for preprocessing
check	Check whether codons split correctly

**Value**

Boolean vector of whether each sequence has an in-frame PTC

---

testPS	<i>Performs PS (parsimony score) test on switch data</i>
--------	--

---

### 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 bootstrap 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:
igphym1 <- "~/apps/igphym1/src/igphym1"
data(ExampleAirr)
ExampleAirr$sample_id <- sample(ExampleAirr$sample_id)
clones <- formatClones(ExampleAirr, trait="sample_id")
btrees <- findSwitches(clones[1:2], bootstraps=10, nproc=1,
  igphym1=igphym1, 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,
  dropzeroes = TRUE,
  bylineage = FALSE,
  pseudocount = 0,
  from = NULL,
  to = NULL,
  permuteAll = FALSE,
  alternative = c("two.sided", "greater", "less")
)
```

**Arguments**

switches	Data frame from findSwitches
dropzeroes	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 bootstrap 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:
igphym1 <- "~/apps/igphym1/src/igphym1"
data(ExampleAirr)
ExampleAirr$sample_id = sample(ExampleAirr$sample_id)
clones = formatClones(ExampleAirr, trait="sample_id")
btrees = findSwitches(clones[1:2], bootstraps=100, nproc=1,
  igphym1=igphym1, 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,
  dropzeroes = 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?
dropzeroes	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 bootstrap 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:
igphym1 <- "~/apps/igphym1/src/igphym1"
data(ExampleAirr)
ExampleAirr$sample_id = sample(ExampleAirr$sample_id)
clones = formatClones(ExampleAirr, trait="sample_id")
btrees = findSwitches(clones[1:2], bootstraps=10, nproc=1,
  igphym1=igphym1, trait="sample_id")
testSP(btrees$switches)

## End(Not run)
```

---

TimeTrees

*Example Ig lineage trees sampled over time.*

---

**Description**

Same as ExampleClones but with timepoint as a trait value

**Usage**

TimeTrees

**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**

[TimeTrees](#)

---

`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

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

### 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)
```

---

writeCloneSequences	<i>Write the sequences used in tree building to a fasta format. If there are more than one tree in airrClone output the sequence id will be followed by "\clone_id".</i>
---------------------	--

---

### Description

writeCloneSequences Exports the sequences used in tree building.

### Usage

```
writeCloneSequences(clones, file)
```

### Arguments

clones	tibble airrClone objects, the output of <a href="#">formatClones</a>
file	The file path and name of where the sequences will be saved

---

writeLineageFile	<i>Write lineage file for IgPhyML use</i>
------------------	---

---

### 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
...	additional arguments to be passed

**Value**

Name of created lineage file.

---

write\_clones\_to\_xmls *Wrapper to write multiple clones to XML files*

---

**Description**

Wrapper to write multiple clones to XML files

**Usage**

```
write_clones_to_xmls(
  data,
  id,
  trees = NULL,
  time = NULL,
  trait = NULL,
  template = NULL,
  outfile = NULL,
  replacements = NULL,
  trait_list = NULL,
  mcmc_length = 1e+06,
  log_every = 1000,
  include_germline_as_root = FALSE,
  include_germline_as_tip = FALSE,
  germline_range = c(-10000, 10000),
  tree_states = FALSE,
  start_edge_length = 100,
  start_date = NULL,
  max_start_date = NULL,
```

```

    germline_trait_value = "?",
    ...
)

```

### Arguments

data	a list of airrClone objects
id	identifer for this analysis
trees	optional list of starting trees, either phylo objects or newick strings
time	name of column representing sample time
trait	name of column representing a trait
template	XML template
outfile	output file path prefix
replacements	list of additional replacements to make in the template
trait_list	list of all possible trait values
mcmc_length	number of MCMC iterations
log_every	frequency of states logged. auto will divide mcmc_length by log_target
include_germline_as_root	include germline in analysis as root?
include_germline_as_tip	include germline in analysis as tip?
germline_range	possible date range of germline
tree_states	use states in the starting tree?
start_edge_length	edge length to use for all branches in starting tree
start_date	starting date to use as prior, in forward time
max_start_date	max starting date to use as prior, in forward time
germline_trait_value	trait value for germline, default '?' for ambiguous
...	additional arguments for XML writing functions

### Value

File paths of the written XML files

---

write_clone_to_xml	<i>Takes an airr clone object and template and writes a BEAST2 XML file</i>
--------------------	---

---

### Description

Takes an airr clone object and template and writes a BEAST2 XML file

### Usage

```
write_clone_to_xml(  
  clone,  
  file,  
  id,  
  time = NULL,  
  trait = NULL,  
  trait_data_type = NULL,  
  template = NULL,  
  mcmc_length = 1e+06,  
  log_every = 1000,  
  replacements = NULL,  
  include_germline_as_root = FALSE,  
  include_germline_as_tip = FALSE,  
  germline_range = c(-10000, 10000),  
  tree = NULL,  
  trait_list = NULL,  
  log_every_trait = 10,  
  tree_states = FALSE,  
  start_edge_length = 100,  
  start_date = NULL,  
  max_start_date = NULL,  
  germline_trait_value = "?",  
  ...  
)
```

### Arguments

clone	an airrClone object
file	output file path
id	unique identifier for this analysis
time	name of column representing sample time
trait	name of column representing a trait
trait_data_type	optional data type for the trait
template	XML template

<code>mcmc_length</code>	number of MCMC iterations
<code>log_every</code>	frequency of states logged. auto will divide <code>mcmc_length</code> by <code>log_target</code>
<code>replacements</code>	list of additional replacements to make in the template
<code>include_germline_as_root</code>	include germline in analysis as root?
<code>include_germline_as_tip</code>	include germline in analysis as tip?
<code>germline_range</code>	possible date range of germline
<code>tree</code>	starting tree, either a phylo object or a newick string
<code>trait_list</code>	list of all possible trait values
<code>log_every_trait</code>	frequency of trait states logged relative to <code>log_every</code>
<code>tree_states</code>	use states in the starting tree?
<code>start_edge_length</code>	edge length to use for all branches in starting tree
<code>start_date</code>	starting date to use as prior, in forward time
<code>max_start_date</code>	max starting date to use as prior, in forward time
<code>germline_trait_value</code>	trait value for germline, default '?' for ambiguous
<code>...</code>	additional arguments for XML writing functions

**Value**

File path of the written XML file

# Index

## \* datasets

- BiopsyTrees, 5
  - ExampleAirr, 32
  - ExampleAirrTyCHE, 33
  - ExampleClones, 34
  - ExampleDbChangeo, 34
  - ExampleMixedClones, 35
  - ExampleMixedDb, 36
  - IsotypeTrees, 56
  - TimeTrees, 84
- airrClone, 31, 43, 59, 75, 76
- airrClone (airrClone-class), 4
- airrClone-class, 4
- BiopsyTrees, 5, 5
- bootstrapTrees, 5
- buildBeast, 7
- buildClonalGermline, 9, 11
- buildGermline, 9, 10, 10, 25
- buildIgphyl, 12, 55
- buildPhylo, 13, 55
- buildPML, 14, 55
- buildPratchet, 16, 55
- buildRAXML, 17, 55
- calcRF, 18
- checkDivergence, 19, 55
- collapseNodes, 20
- colorTrees, 20
- condenseTrees, 21
- correlationTest, 22, 74, 75
- create\_alignment, 26
- create\_height\_prior, 26
- create\_max\_height\_prior, 27
- create\_MRCA\_prior\_germline, 27
- create\_MRCA\_prior\_observed, 28
- create\_root\_freqs, 28
- create\_starting\_tree, 29
- create\_traitset, 29
- createGermlines, 10, 23, 23, 25
- dfToFasta, 30
- downsampleClone, 31
- dowser, 32
- ExampleAirr, 32, 35
- ExampleAirrTyCHE, 33
- ExampleClones, 33, 34, 34, 35
- ExampleDbChangeo, 33, 34
- ExampleMixedClones, 35
- ExampleMixedDb, 36
- exportTrees, 38
- findSwitches, 38, 43, 55, 60, 66, 70, 80, 82, 84
- formatClones, 4, 6, 38–40, 40, 44, 52, 54, 55, 59, 86
- getAllSeqs, 43
- getBootstraps, 44
- getDivergence, 45
- getGermline, 46, 46, 77, 78
- getNodeSeq, 43, 47
- getPalette, 20, 21, 47
- getSeq, 48
- getSkylines, 49, 64
- getSubclones, 50
- getSubTaxa, 51
- getTimeTrees, 8, 51
- getTimeTreesIterate, 53
- getTrees, 19–21, 31, 40, 43, 47, 48, 53, 54, 60, 64, 66, 70, 74, 76, 77
- IsotypeTrees, 56, 56
- makeAirrClone, 41, 43, 57
- makeModelFile, 59, 70
- makeSkyline, 60
- maskCodons, 61, 63
- maskSequences, 62, 62

plotSkylines, 64  
plotTrees, 21, 40, 48, 64, 85

readBEAST, 53, 64, 66  
readFasta, 67  
readIMGT, 9, 24, 68  
readLineages, 68  
readModelFile, 60, 69  
reconIgPhyML, 70  
rerootTree, 71  
resolveLightChains, 72  
resolvePolytomies, 73  
runCorrelationTest, 74

sampleCloneMultiGroup, 75  
sampleClones, 76  
scaleBranches, 76  
stitchRegions, 77, 77  
stitchVDJ, 10, 11, 25, 77, 78, 78  
stopCodonCheck, 79

testPS, 40, 80, 82, 84  
testSC, 40, 80, 81, 84  
testSP, 40, 80, 82, 82  
TimeTrees, 84, 84  
treesToPDF, 85

write\_clone\_to\_xml, 89  
write\_clones\_to\_xmls, 87  
writeCloneSequences, 86  
writeLineageFile, 86