

Package ‘genealogicalSorting’

May 8, 2009

Title genealogical sorting index for quantifying the degree of exclusive ancestry of labeled groups on a rooted genealogy

Version 0.9

Author Adam L. Bazinet, Daniel S. Myers, Pratik Khatavkar

Maintainer Adam Bazinet <pknut777@umiacs.umd.edu>

Depends ape

Suggests Rmpi, snow

Description The genealogical sorting index is a metric for quantifying the common ancestry of groups of taxa on a phylogenetic tree.

License GPL (>= 2)

URL <http://www.genealogicalsorting.org/>

R topics documented:

allKidsSameGroup	1
clientFunc	2
computeIndexHelper	3
createAssignment	3
createTaxaHash	4
doParallelPerm	4
doSerialPerm	5
findChildren	6
generatePermutedAssignments	6
getAllInternalNodes	7
getGroupFromAssignment	7
getNumChildren	8
getNumGroup	8
getNumInternalNodes	9
getNumLeafNodes	9
groupStringsToIntHash	10
groupStringsToInts	10
groupTableToVector	11
gsi	11
hasNGroupKids	12

initRandomSeed	13
isLeaf	13
masterGroupInts	14
masterGroupOrder	14
multiTreeAnalysis	15
permutationTest	16
randomizeAssignments	17
readAssignmentFile	17
readAssignmentTable	18
readWeightedNexus	19
readWeightedTree	19
reducePermmmap	20
singleTreeAnalysis	20
writeAssignmentFile	21

<code>allKidsSameGroup</code>	<i>All Child Nodes Same Group (INTERNAL)</i>
-------------------------------	--

Description

Returns true if all children of a given node are of the given group.

Usage

```
allKidsSameGroup(tree, assignments, parentid, group)
```

Arguments

<code>tree</code>	An object of type phylo.
<code>assignments</code>	A list of groups for the taxa in the tree.
<code>parentid</code>	ID of the parent node in the tree to search from.
<code>group</code>	A string containing the group to check.

Value

Returns TRUE if all children of parentid are of the given group.

Author(s)

ALB

clientFunc	<i>Helper Function for doSerialPerm, doParallelPerm (INTERNAL)</i>
------------	--

Description

Function run on each client node in a permutation calculation.

Usage

```
clientFunc(permmap, tree, assignments, intassignments, nperms, origindexes)
```

Arguments

permmap	A vector of permuted assignments.
tree	A tree object of class "phylo".
assignments	A vector of groups.
intassignments	
nperms	Number of permutations to do.
origindexes	An array containing the values of <i>gsi</i> in the original assignments for each group.

Value

nbetter	
indices	

Returns a matrix of all calculated *gsi* values.

Author(s)

ALB & DSM

computeIndexHelper	<i>Helper Function for computeIndex (INTERNAL)</i>
--------------------	--

Description

A helper function for computeIndex that does most of the work.

Usage

```
computeIndexHelper(tree, group, node, assignments, ngroup)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>group</code>	The group of interest.
<code>node</code>	Index of the root node (typically -1)
<code>assignments</code>	A vector of groups.
<code>ngroup</code>	The number of occurrences of this group in the tree.

Value

Returns the genealogical sorting index.

Author(s)

ALB & DSM

`createAssignment` *Create a Subset of the Master Assignments (INTERNAL)*

Description

Creates a subset of the master assignments containing the taxa provided.

Usage

```
createAssignment(assignmentsTable, taxa)
```

Arguments

<code>assignmentsTable</code>	The master assignment table.
<code>taxa</code>	A vector of taxa.

Value

Returns a vector of groups for the taxa given.

Author(s)

ALB

<code>createTaxaHash</code>	<i>Creates a Hash of the Taxon Name to its Position in the Given Table (INTERNAL)</i>
-----------------------------	---

Description

Creates an array where the value of the taxon name is its position in the assignment table.

Usage

```
createTaxaHash(assignmentsTable)
```

Arguments

`assignmentsTable`
A table assigning taxa to groups.

Value

Returns an array of taxa and their positions in the assignment table.

Author(s)

ALB

<code>doParallelPerm</code>	<i>Computes Permutations Using SNOW (INTERNAL)</i>
-----------------------------	--

Description

Does distributed permutation testing using the snow package and the Rmpi package. Requires a LAM environment to be set up.

Usage

```
doParallelPerm(cluster, tree, assignments, intassignments, permmap, nperms, nprocs, groups)
```

Arguments

`cluster` A pre-initialized snow cluster object.
`tree` A tree object of class "phylo".
`assignments` A vector of groups.
`intassignments` A vector of integer group assignments.
`permmap` A vector of permuted assignments.
`nperms` The number of permutations to run.
`nprocs` The number of processors to distribute over.
`groups` In a multiTreeAnalysis, this is the master group ordering.

Value

<code>groups</code>	An ordering of groups that were tested.
<code>indexes</code>	Original <i>gsi</i> indexes for the groups.
<code>pvalsbetter</code>	<i>P</i> -values for the groups.
<code>nperms</code>	The number of permutations run.

Author(s)

ALB

<code>doSerialPerm</code>	<i>Computes Permutations on a Single Processor (INTERNAL)</i>
---------------------------	---

Description

Does permutation testing on a single processor.

Usage

```
doSerialPerm(tree, assignments, intassignments, permmap, nperms, groups)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>assignments</code>	A vector of groups.
<code>intassignments</code>	A vector of integer group assignments.
<code>permmap</code>	A vector of permuted assignments.
<code>nperms</code>	The number of permutations to run.
<code>groups</code>	In a multiTreeAnalysis, this is the master group ordering.

Value

<code>groups</code>	An ordering of groups that were tested.
<code>indexes</code>	Original <i>gsi</i> indexes for the groups.
<code>pvalsbetter</code>	<i>P</i> -values for the groups.
<code>nperms</code>	The number of permutations run.

Author(s)

ALB

<code>findChildren</code>	<i>Find Child Nodes of a Parent (INTERNAL)</i>
---------------------------	--

Description

Takes in a tree object of class "phylo" and a parent ID and returns a list containing the IDs of the children.

Usage

```
findChildren(tree, parentid)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>parentid</code>	A parent ID representing a node in the tree.

Value

A list containing the IDs of the children.

<code>IDs</code>	IDs of the child nodes
------------------	------------------------

Author(s)

ALB & DSM

<code>generatePermutedAssignments</code>	<i>Generates Permuted Assignments (INTERNAL)</i>
--	--

Description

Generates a number of permuted assignments equal to `nreps + nprocs`.

Usage

```
generatePermutedAssignments(assignmentFile, trees, nreps, nprocs)
```

Arguments

<code>assignmentFile</code>	Path to the assignment file on disk.
<code>trees</code>	A tree object of class "phylo" and perhaps "multiPhylo". If "multi.tree", a more complex algorithm is used to generate the assignments.
<code>nreps</code>	The number of permutations that will be run.
<code>nprocs</code>	The number of processors over which computation will be distributed.

Value

Returns a structure containing the permuted assignments.

Author(s)

ALB

`getAllInternalNodes` *Counts Non-leaf Nodes (INTERNAL)*

Description

Returns the number of children of the parent node that are not leaves.

Usage

```
getAllInternalNodes(tree, parentid, assignments)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>parentid</code>	The node to start from.
<code>assignments</code>	A vector of groups.

Value

The number of children of the parent node that are not leaves.

Author(s)

ALB & DSM

`getGroupFromAssignment`
Get the Group of an Element in a Assignment (INTERNAL)

Description

Gets the group of an element in 'assignments'.

Usage

```
getGroupFromAssignment(assignments, node)
```

Arguments

<code>assignments</code>	A vector of groups.
<code>node</code>	The index into the assignments vector.

Value

The group of interest.

Author(s)

ALB & DSM

getNumChildren	<i>Number of Children of a Node (INTERNAL)</i>
----------------	--

Description

Returns the number of children of a node.

Usage

```
getNumChildren(tree, parentid)
```

Arguments

<code>tree</code>	An object of class "phylo".
<code>parentid</code>	A parent ID representing a node in the tree.

Value

The number of children, an integer.

Author(s)

ALB & DSM

getNumGroup	<i>Get Number of Child Nodes Matching a Given Group (INTERNAL)</i>
-------------	--

Description

Returns the number of children whose group matches the function argument.

Usage

```
getNumGroup(tree, assignments, parentid, group)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>assignments</code>	A vector of groups.
<code>parentid</code>	The node to start from.
<code>group</code>	The group to search for.

Value

Number of matching children.

Author(s)

ALB & DSM

`getNumInternalNodes` *Counts Non-leaf Nodes (INTERNAL)*

Description

Returns the number of children of the parent node that are not leaves and match the given group.

Usage

```
getNumInternalNodes(tree, parentid, assignments, group)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>parentid</code>	The node to start from.
<code>assignments</code>	A vector of groups.
<code>group</code>	The group for which to calculate.

Value

The number of children of the parent node that are not leaves and match the given group.

Author(s)

ALB & DSM

`getNumLeafNodes` *Counts Leaf Nodes (INTERNAL)*

Description

Returns the number of children of the parent node that are leaf nodes.

Usage

```
getNumLeafNodes(tree, parentid)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>parentid</code>	The node to start from.

Value

The number of children of the parent node that are leaves.

Author(s)

ALB

groupStringsToIntHash

Assign Groups an Integer Value (INTERNAL)

Description

Assign each group in the assignment vector a sequential integer value.

Usage

```
groupStringsToIntHash(assignments)
```

Arguments

assignments A vector of groups.

Value

A hash mapping group strings to integers.

Author(s)

ALB

groupStringsToInts

Assign Groups an Integer Value (INTERNAL)

Description

Assign each group in the assignment vector a sequential integer value.

Usage

```
groupStringsToInts(assignments)
```

Arguments

assignments A vector of groups.

Value

A vector of integer assignments.

Author(s)

ALB

`groupTableToVector` *Convert Group Table to Vector (INTERNAL)*

Description

Creates a vector of groups that correspond to the taxa in the given tree.

Usage

```
groupTableToVector(grouptable, tree)
```

Arguments

`grouptable` A table which assigns taxa to groups.
`tree` A tree object of class "phylo".

Value

Returns an assignment vector.

Author(s)

ALB

`gsi` *Calculates the Genealogical Sorting Index*

Description

Calculates the genealogical sorting index for a given group in a given tree.

Usage

```
gsi(tree, group, assignments)
```

Arguments

`tree` A tree object of class "phylo".
`group` The group to calculate *gsi* for. If group=0, calculate *gsi* for all groups.
`assignments` A vector of groups.

Value

Returns the genealogical sorting index, a value in the interval [0, 1].

Author(s)

ALB, DSM, PPK

Examples

```
# generate a tree
tree <- rtree(10)

# generate random assignments
assignments <- randomizeAssignments(tree, c("group1", "group2"))
print(assignments)

# compute gsi for group1
gsi(tree,"group1",assignments)

# compute gsi for all groups
gsi(tree,0,assignments)
```

hasNGroupKids	<i>Helper Function for computeIndex (INTERNAL)</i>
---------------	--

Description

Returns true if the given node has nchild children of the given group.

Usage

```
hasNGroupKids(tree, group, assignments, node, nchild)
```

Arguments

tree	A tree object of class "phylo".
group	The group of interest.
assignments	A vector of groups.
node	Parent node to start from.
nchild	Number of children to test for.

Value

Boolean, true if given node has nchild children of the given group.

Author(s)

ALB & DSM

<code>initRandomSeed</code>	<i>Seeds the Random Number Generator (INTERNAL)</i>
-----------------------------	---

Description

Seeds the random number generator from the current date and time. Important if you want different results each time you start an R session.

Usage

```
initRandomSeed()
```

Arguments**Value****Author(s)**

ALB

<code>isLeaf</code>	<i>Check if a Node is a Terminal Node (INTERNAL)</i>
---------------------	--

Description

Check if given node is a leaf node.

Usage

```
isLeaf(tree, id)
```

Arguments

<code>tree</code>	A tree object of class "phylo".
<code>id</code>	Node to check.

Value

Boolean, true if leaf node.

Author(s)

ALB & DSM

masterGroupInts	<i>Assign Groups an Integer Value (INTERNAL)</i>
-----------------	--

Description

Convert group assignments from strings to integers using the master group integer mappings.

Usage

```
masterGroupInts(masterassignments, treeassignments)
```

Arguments

masterassignments
A vector of groups in the master ordering.

treeassignments
A tree-specific vector of groups.

Value

A vector of integer assignments.

Author(s)

ALB

masterGroupOrder	<i>Reorder Groups (INTERNAL)</i>
------------------	----------------------------------

Description

Reorder groups according to a master ordering.

Usage

```
masterGroupOrder(masterassignments, treeassignments)
```

Arguments

masterassignments
A vector of unique groups in the master ordering.

treeassignments
A tree-specific vector of unique groups.

Value

A vector of reordered group assignments.

Author(s)

ALB

multiTreeAnalysis *Computes gsi and P-values for the Groups in a Set of Trees*

Description

Computes *gsi* and *P*-values for the groups in a set of trees. Also calculates ensemble *gsi* and ensemble *P*-values.

Usage

```
multiTreeAnalysis(trees, assignmentFile, nperms, nprocs,
                  weights = array(1, length(trees)))
```

Arguments

trees	An object of class "phylo" and class "multiPhylo".
assignmentFile	The path to an assignment file.
nperms	The number of permutations to run.
nprocs	The number of processors to distribute computation over. If nprocs=1, the computation is done locally.
weights	An array of how the trees should be weighted when calculating ensemble <i>gsi</i> statistics.

Value

singleTreeData	A list containing <i>gsi</i> and <i>P</i> -values for each group in each tree.
multiTreeData	A list containing ensemble <i>gsi</i> and <i>P</i> -values for each group calculated across all trees.
nperms	The number of permutations run.

Author(s)

ALB

Examples

```
# generate a multi-tree
trees <- rmtree(2,10)

# generate random assignments
assignments <- randomizeAssignments(trees[[1]], c("group1", "group2"))

# write assignments to disk
assignmentFile <- tempfile("tempassignments")
writeAssignmentFile(trees[[1]], assignments, assignmentFile)

# perform analysis using assignment file already created on disk
results <- multiTreeAnalysis(trees, assignmentFile, 1000, 1)
```

permutationTest	<i>Permutation Test the gsi statistic</i>
-----------------	---

Description

Calculates the genealogical sorting index n perms times, randomizing the group assignments each iteration.

Usage

```
permutationTest(tree, group, assignments, nperms)
```

Arguments

tree	A tree object of class "phylo".
group	The group of interest. If group=0, calculate a P -value for all groups.
assignments	A vector of groups.
nperms	Number of times to permute and recalculate <i>gsi</i> .

Value

Returns the probability of seeing the observed genealogical sorting index value or greater.

Author(s)

ALB & PPK

Examples

```
# generate a tree
tree <- rtree(10)

# generate a random assignment vector
assignments <- randomizeAssignments(tree, c("group1", "group2"))

# permutation test group1
permutationTest(tree,"group1",assignments,1000)

# permutation test all groups
permutationTest(tree,0,assignments,1000)
```

`randomizeAssignments` *Generate a Random Assignment Vector*

Description

Generate a random assignment vector from tree labels to groups.

Usage

```
randomizeAssignments(tree, groups)
```

Arguments

`tree` A tree object of class "phylo".
`groups` A vector of groups to randomly allocate.

Value

A randomly generated assignment from tree labels to groups. Attempts to assign at least two individuals to each group.

Author(s)

ALB & DSM

Examples

```
# read in a tree
tree <- read.tree("~/trees/my_tree.phy")

# generate a random assignment vector
assignments <- randomizeAssignments(tree, c("group1", "group2"))
```

`readAssignmentFile` *Read an Assignment File from Disk*

Description

Reads an assignment file from disk.

Usage

```
readAssignmentFile(assignmentFile, tree)
```

Arguments

`assignmentFile` Path to the assignment file.
`tree` A tree object to map to.

Value

Returns an assignment vector.

Author(s)

ALB & DSM

Examples

```
# read in a tree
tree <- read.tree("~/trees/my_tree.phy")

# read assignment file from disk
assignments <- readAssignmentFile("~/trees/my_tree_assignments", tree)
print(assignments)
```

`readAssignmentTable` *Reads an Assignment File on Disk into a Table (INTERNAL)*

Description

Reads an assignment file from disk and returns the table directly.

Usage

```
readAssignmentTable(assignmentsFile)
```

Arguments

`assignmentsFile`
Path to the assignment file.

Value

Returns an assignment table.

Author(s)

ALB

`readWeightedNexus` *Read Weighted Nexus File*

Description

Reads a weighted nexus tree file from disk and returns the trees and weights, respectively. The function will automatically handle weights in fractional or decimal format, and will combine duplicate trees and add their weights together appropriately.

Usage

```
readWeightedNexus(file, tree.names = NULL)
```

Arguments

`file` Path to the tree file.
`tree.names` An optional list of names for the trees.

Value

`trees` An object of type phylo containing one or more trees.
`weights` A list of weights whose length equals the number of trees.

Author(s)

ALB

`readWeightedTree` *Reads a Weighted Phylip or Newick Tree File on Disk*

Description

Reads a weighted phylip or newick tree file on disk and returns the trees and weights, respectively. Automatically combines duplicate trees in the file.

Usage

```
readWeightedTree(file)
```

Arguments

`file` Path to the tree file.

Value

`trees` An object of type "phylo" and possibly "multiPhylo" containing one or more trees.
`weights` A list of integer weights whose length equals the number of trees.

Author(s)

ALB

reducePermmmap	<i>Reduce Permuted Assignments (INTERNAL)</i>
----------------	---

Description

Goes through each list in the array and removes groups that are not in the given tree.

Usage

```
reducePermmmap(permmmap, assignmentsHash, tree)
```

Arguments

permmmap	An array of permuted assignments.
assignmentsHash	An array which assigns taxa names to assignment positions.
tree	An object of type 'phylo'.

Value

Returns a new set of permuted assignments.

Author(s)

ALB

singleTreeAnalysis	<i>Compute gsi and P-values for the Groups in a Single Tree</i>
--------------------	---

Description

Computes the genealogical sorting index and *P*-values for the groups in a single tree.

Usage

```
singleTreeAnalysis(tree, assignmentFile, nperms, nprocs)
```

Arguments

tree	A tree object of class "phylo".
assignmentFile	The path to an assignment file.
nperms	The number of permutations to run.
nprocs	The number of processors to distribute computation over. If nprocs=1, the computation is done locally.

Value

groups	An ordering of groups that were tested.
indexes	Original <i>gsi</i> indexes for the groups.
pvals	<i>P</i> -values for the groups.
nperms	The number of permutations run.

Author(s)

ALB

Examples

```
# generate a tree
tree <- rtree(10)

# generate random assignments
assignments <- randomizeAssignments(tree, c("group1", "group2"))

# write assignments to disk
assignmentFile <- tempfile("tempassignments")
writeAssignmentFile(tree, assignments, assignmentFile)

# perform analysis using an assignment file already created on disk
results <- singleTreeAnalysis(tree, assignmentFile, 1000, 1)
```

writeAssignmentFile *Write an Assignment File to Disk*

Description

Writes an assignment file to disk.

Usage

```
writeAssignmentFile(tree, assignments, fileName)
```

Arguments

tree	A tree object of class "phylo".
assignments	A list of groups.
fileName	The file name to write to.

Value**Author(s)**

ALB

Examples

```
# generate a tree file
tree <- rtree(10)

# generate random assignments
assignments <- randomizeAssignments(tree, c("group1", "group2"))

# write assignments to disk
assignmentFile <- tempfile("tempassignments")
writeAssignmentFile(tree, assignments, assignmentFile)
```

Index

- *Topic **cluster**
 - gsi, 11
 - multiTreeAnalysis, 15
 - permutationTest, 16
 - singleTreeAnalysis, 20
- *Topic **file**
 - readAssignmentFile, 17
 - readWeightedNexus, 19
 - readWeightedTree, 19
 - writeAssignmentFile, 21
- *Topic **internal**
 - allKidsSameGroup, 1
 - clientFunc, 2
 - computeIndexHelper, 3
 - createAssignment, 3
 - createTaxaHash, 4
 - doParallelPerm, 4
 - doSerialPerm, 5
 - findChildren, 6
 - generatePermutedAssignments, 6
 - getAllInternalNodes, 7
 - getGroupFromAssignment, 7
 - getNumChildren, 8
 - getNumGroup, 8
 - getNumInternalNodes, 9
 - getNumLeafNodes, 9
 - groupStringsToIntHash, 10
 - groupStringsToInts, 10
 - groupTableToVector, 11
 - hasNGroupKids, 12
 - initRandomSeed, 13
 - isLeaf, 13
 - masterGroupInts, 14
 - masterGroupOrder, 14
 - readAssignmentTable, 18
 - reducePermmmap, 20
- *Topic **misc**
 - randomizeAssignments, 17
- *Topic **tree**
 - masterGroupOrder, 14
- allKidsSameGroup, 1
- clientFunc, 2
- computeIndexHelper, 3
- createAssignment, 3
- createTaxaHash, 4
- doParallelPerm, 4
- doSerialPerm, 5
- findChildren, 6
- generatePermutedAssignments, 6
- getAllInternalNodes, 7
- getGroupFromAssignment, 7
- getNumChildren, 8
- getNumGroup, 8
- getNumInternalNodes, 9
- getNumLeafNodes, 9
- groupStringsToIntHash, 10
- groupStringsToInts, 10
- groupTableToVector, 11
- gsi, 11
- hasNGroupKids, 12
- initRandomSeed, 13
- isLeaf, 13
- masterGroupInts, 14
- masterGroupOrder, 14
- multiTreeAnalysis, 15
- permutationTest, 16
- randomizeAssignments, 17
- readAssignmentFile, 17
- readAssignmentTable, 18
- readWeightedNexus, 19
- readWeightedTree, 19
- reducePermmmap, 20
- singleTreeAnalysis, 20
- writeAssignmentFile, 21