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 Khatakar

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
Description

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

Usage

allKidsSameGroup(tree, assignments, parentid, group)

Arguments

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

Value

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

Author(s)

ALB
clientFunc

**Helper Function for doSerialPerm, doParallelPerm (INTERNAL)**

**Description**

Function run on each client node in a permutation calculation.

**Usage**

```r
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 gsi in the original assignments for each group.

**Value**

- `nbetter`
- `indices`

Returns a matrix of all calculated gsi values.

**Author(s)**

ALB & DSM

---

computeIndexHelper

**Helper Function for computeIndex (INTERNAL)**

**Description**

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

**Usage**

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

- `assignmentsTable`: The master assignment table.
- `taxa`: A vector of taxa.

Value

Returns a vector of groups for the taxa given.

Author(s)

ALB
**createTaxaHash**  
*Creates a Hash of the Taxon Name to its Position in the Given Table (INTERNAL)*

**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

---

**doParallelPerm**  
*Computes Permutations Using SNOW (INTERNAL)*

**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.
### doSerialPerm
*Computes Permutations on a Single Processor (INTERNAL)*

#### Description
Does permutation testing on a single processor.

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

#### Arguments
- **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.
- **groups**: In a multiTreeAnalysis, this is the master group ordering.

#### Value
- **groups**: An ordering of groups that were tested.
- **indexes**: Original gsi indexes for the groups.
- **pvalsbetter**: $P$-values for the groups.
- **nperms**: The number of permutations run.

#### Author(s)
ALB
**findChildren**  
*Find Child Nodes of a Parent (INTERNAL)*

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

**Usage**

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

**Arguments**
- **tree**  
  A tree object of class "phylo".
- **parentid**  
  A parent ID representing a node in the tree.

**Value**
A list containing the IDs of the children.

**IDs**  
IDs of the child nodes

**Author(s)**
ALB & DSM

---

**generatePermutedAssignments**  
*Generates Permuted Assignments (INTERNAL)*

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

**Usage**

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

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

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
- tree: A tree object of class "phylo".
- parentid: The node to start from.
- assignments: 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
- assignments: A vector of groups.
- node: The index into the assignments vector.
**getNumChildren**

**Value**

The group of interest.

**Author(s)**

ALB & DSM

---

**getDescription**

**Description**

Returns the number of children of a node.

**Usage**

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

**Arguments**

- `tree`: An object of class "phylo".
- `parentid`: A parent ID representing a node in the tree.

**Value**

The number of children, an integer.

**Author(s)**

ALB & DSM

---

**getNumGroup**

**Description**

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

**Usage**

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

**Arguments**

- `tree`: A tree object of class "phylo".
- `assignments`: A vector of groups.
- `parentid`: The node to start from.
- `group`: The group to search for.
getValue

Number of matching children.

Author(s)

ALB & DSM

ggetNumInternalNodes  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

ggetNumInternalNodes(tree, parentid, assignments, group)

Arguments

tree A tree object of class "phylo".
parentid The node to start from.
assignments A vector of groups.
group 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

ggetNumLeafNodes  Counts Leaf Nodes (INTERNAL)

Description

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

Usage

ggetNumLeafNodes(tree, parentid)

Arguments

tree A tree object of class "phylo".
parentid 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
```plaintext
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
```plaintext
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

```r
# 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**

*Helper Function for computeIndex (INTERNAL)*

**Description**

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

**Usage**

```r
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
initRandomSeed  
*Seeds the Random Number Generator* *(INTERNAL)*

**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**

```r
initRandomSeed()
```

**Arguments**

**Value**

**Author(s)**

ALB

---

isLeaf  
*Check if a Node is a Terminal Node* *(INTERNAL)*

**Description**

Check if given node is a leaf node.

**Usage**

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

**Arguments**

- `tree`  
  A tree object of class "phylo".
- `id`  
  Node to check.

**Value**

Boolean, true if leaf node.

**Author(s)**

ALB & DSM
masterGroupInts  

Assign Groups an Integer Value (INTERNAL)

**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  

Reorder Groups (INTERNAL)

**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

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 gsi statistics.

Value

singleTreeData A list containing gsi and P-values for each group in each tree.
multiTreeData A list containing ensemble gsi and P-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  

Permutation Test the gsi statistic

Description

Calculates the genealogical sorting index nperms 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 gsi.

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
```r
# 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

```r
# 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

```r
readAssignmentTable(assignmentsFile)
```

Arguments

- `assignmentsFile`
  
  Path to the assignment file.

Value

Returns an assignment table.

Author(s)

ALB
readWeightedNexus  \hspace{2cm} \textit{Read Weighted Nexus File}

\textbf{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.

\textbf{Usage}

\begin{verbatim}
readWeightedNexus(file, tree.names = NULL)
\end{verbatim}

\textbf{Arguments}

- \texttt{file} \hspace{1cm} Path to the tree file.
- \texttt{tree.names} \hspace{1cm} An optional list of names for the trees.

\textbf{Value}

- \texttt{trees} \hspace{1cm} An object of type phylo containing one or more trees.
- \texttt{weights} \hspace{1cm} A list of weights whose length equals the number of trees.

\textbf{Author(s)}

ALB

---

readWeightedTree  \hspace{2cm} \textit{Reads a Weighted Phylib or Newick Tree File on Disk}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
readWeightedTree(file)
\end{verbatim}

\textbf{Arguments}

- \texttt{file} \hspace{1cm} Path to the tree file.

\textbf{Value}

- \texttt{trees} \hspace{1cm} An object of type "phylo" and possibly "multiPhylo" containing one or more trees.
- \texttt{weights} \hspace{1cm} A list of integer weights whose length equals the number of trees.

\textbf{Author(s)}

ALB
reducePermmap  
*Reduce Permuted Assignments (INTERNAL)*

**Description**

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

**Usage**

```r
reducePermmap(permmap, assignmentsHash, tree)
```

**Arguments**

- `permmap`: 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  
*Compute gsi and P-values for the Groups in a Single Tree*

**Description**

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

**Usage**

```r
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 *gsi* indexes for the groups.
- **pvals**: *P*-values for the groups.
- **nperms**: The number of permutations run.

Author(s)

- ALB

Examples

```r
# 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

```r
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

```r
# 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)
```