Package 'HMPTrees'

October 12, 2022

Type Package

Title Statistical Object Oriented Data Analysis of RDP-Based Taxonomic Trees from Human Microbiome Data

Version 1.4

Date 2017-07-05

Author Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Maintainer Berkley Shands <rpackages@biorankings.com>

Depends R (>= 3.0.0)

Imports ape, HMP, dirmult, doParallel, foreach, parallel, stats, graphics

Description Tools to model, compare, and visualize populations of taxonomic tree objects.

License Apache License (== 2.0)

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-05 22:08:45 UTC

R topics documented:

HMPTrees-package *Object Oriented Data Analysis of Taxonomic Trees*

Description

Object Oriented Data Analysis (OODA) methods to analyze Human Microbiome taxonomic trees directly. We provide tools to model, compare, and visualize populations of taxonomic trees.

Details

HMP metagenomic sequences in a sample can be represented as a rooted taxonomic tree. Using supervised taxonomic methods a sequence is matched to a hierarchical taxa or taxonomy bins defined in a bacterial-taxonomy library such as, for example, the Ribosomal Database Project (RDP) (Cole, 2005). The supervised taxonomic analysis allows us to represent each sample (set of sequences) by a rooted taxonomic tree where the root corresponds to taxon at the Kingdom level, i.e., bacteria, and the leaves correspond to the taxa at the Genus level, and the width of the edges (paths) between taxonomic levels correspond to the 'abundances' of the descending taxon.

In particular, we combine RDP matches by adding RDP values of common taxon, which allows us to provide a measure of taxa abundance weighting on the confidence of each taxa assignment. The resulting taxonomic trees satisfy the following conditions: i) branches closer to the root have higher 'abundance' values than branches closer to leaves, and ii) the sum of the 'abundances' of all descending taxa under a common parent taxon cannot be larger than the 'abundance' of the corresponding parent taxon.

It is important to note that due to how the ape package works the following naming conventions apply to taxa names:

1. Colons cannot be used in the taxa names at all.
2. Each taxa name must be unique - you cannot have two seperate branches both have a child named 'unclassified' for example . (We took the parent name and added a 'U' to the end to signify an unclassified in our data sets)
3. There can only be one top level node. (Bacteria and Archaea cannot both exist unless there is an additional single level above them for example)

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

References

1. Cole JR, Chai B, Farris RJ, Wang Q, Kulam SA, McGarrell DM, Garrity GM, Tiedje JM. The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis. Nucleic Acids Research 2005; 33: D294-D296.
2. P. S. La Rosa, Berkley Shands, Elena Deych, Yanjiao Zhou, Erica Sodergren, George Weinstock, and William D. Shannon, "Object data analysis of taxonomic trees from human microbiome data,"PLoS ONE 7(11): e48996. doi:10.1371/journal.pone.0048996. Nov. 2012.
3. Banks D, Constantine GM. Metric Models for Random Graphs. Journal of Classification 1998; 15: 199-223.
4. Shannon WD, Banks D. Combining classification trees using MLE. Stat Med 1999; 18: 727-740.

`checkTreeValidity` *Check Validity of an RDP-Based Taxonomic Tree*

Description

This function goes through every node in the tree and for each node it checks that the sum of that nodes children are less than or equal to the value of that node.

Usage

```
checkTreeValidity(data, samples = NULL, epsilon = 0.0001, split = ".")
```

Arguments

`data` A data frame in which each column contains the rdp read counts for every taxa given in the row names.

`samples` Deprecated. Only send the columns in data to create.

`epsilon` This value allows for rounding problems or other such small errors in the data such that the (parent + epsilon > sum(children)).

`split` This is the character that separates the taxa levels in the row names.

Value

A boolean vector that indicates the validity of every tree tested.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)

validTree <- checkTreeValidity(saliva[,1, drop=FALSE])
validTree
```

compareTwoDataSets *Likelihood-Ratio-Test Statistics to Compare the Distribution of 2 Sets of RDP-Based Taxonomic Trees*

Description

This functions compares the distribution of two sets of RDP-based taxonomic trees using Likelihood-Ratio-Test statistics and a p-value is computed using permutations.

Usage

```
compareTwoDataSets(data1, data2, numPerms = 1000, parallel = FALSE, cores = 3,
maxSteps=50, delta=10^(-6), numBootStraps = NULL, enableMC = NULL)
```

Arguments

`data1, data2`	Data frames in which each column contains the rdp read counts for every taxa given in the row names.
`numPerms`	The number of permutation tests to run.
`parallel`	When this is 'TRUE' it allows for parallel calculation of the permutations. Requires the package `doParallel`.
`cores`	The number of parallel processes to run if enableMC is 'TRUE'.
`maxSteps`	The maximum number of times to iterate though for the MLE.
`delta`	The minimum threshold of change in f to stop the search for the MLE.
`numBootStraps`	Deprecated. Replaced with numPerms.
`enableMC`	Deprecated. Replaced with parallel.

Details

Note: Both data sets should be standardized to the same number of reads.

We are interested in assessing whether the distributions from two metagenomic populations are the same or different, which is equivalent to evaluating whether their respective parameters are the same or different. The corresponding hypothesis is given as follows:

$$H_{\mathrm{o}} : (g_1^*, \tau_1) = (g_2^*, \tau_2) = (g_0^*, \tau_0) vs H_{\mathrm{A}} : (g_1^*, \tau_1) \neq (g_2^*, \tau_2),$$

where (g_0^*, τ_0) is the unknown common parameter vector. To evaluate this hypothesis we use the likelihood-ratio test (LRT) which is given by,

$$\lambda = -2 \log \left(\frac{L(g_o^*, \tau_o; S_{1n}, S_{2m})}{L(g_1^*, \tau_1; S_{1n}) + L(g_2^*, \tau_2; S_{2m})} \right),$$

where S_{1n} and S_{2m} are the sets containing n and m random samples of trees from each metagenomic population, respectively. We assume that the model parameters are unknown under both the null and alternative hypothesis, therefore, we estimate these using the MLE procedure proposed in La Rosa et al (see reference 2), and compute the corresponding p-value using non-parametric bootstrap.

Value

A p-value for the similarity of the two data sets based on the permutation test.

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Examples

```
data(saliva)
data(stool)

### We use 1 for the number of permutations for computation time
### This value should be at least 1000 for an accurate result
numPerms <- 1

pval <- compareTwoDataSets(saliva, stool, numPerms)
pval
```

`createAndPlot` *Create and Plot a Tree from a Data Set*

Description

This function combines the createTrees and plotTree functions to create and plot a set of trees.

Usage

```
createAndPlot(data, samples = NULL, level = "genus", colors = NULL,
divisions = NULL, main = NULL, sub = "", showTipLabel = TRUE,
showNodeLabel = FALSE, displayLegend = TRUE, onePerPage = FALSE,
split = ".")
```

Arguments

`data`	A data frame in which each column contains the rdp read counts for every taxa given in the row names.
`samples`	Deprecated. Only send the columns in data to plot.
`level`	The depth the tree creation will go down to (kingdom, phylum, class, order, family, genus, species, subspecies).
`colors`	A vector of colors to be applied to the branches in the plot.
`divisions`	A vector of numbers to be used as break points to assign different colors.
`main`	A custom title(s) for the plot(s).
`sub`	A custom subtitle for the plot.
`showTipLabel`	Hides the tip labels if 'FALSE' otherwise it shows all non-zero tip labels.

`showNodeLabel` Hides the interior node labels if 'FALSE' otherwise it shows all non-zero node labels.

`displayLegend` Enables the display of a legend of the branch colors and divisions when 'TRUE'.

`onePerPage` If 'TRUE' one tree will be plotted per page, if 'FALSE' four will be displayed per page.

`split` This is the character that separates the taxa levels in the row names.

Details

Notes:

1. For 'level' k, p, c, o, f, g, s and ss can be used in place of kingdom, phylum, class, order, family, genus, species and subspecies respectively.
2. The values for division should directly relate to the values of your data, i.e. if your data ranges from 0 to 50000 reads you should adjust the divisions to fit your data.

Value

A plot of the tree(s).

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)

### Plots the trees in column 2 and 3 in 'Saliva'
createAndPlot(saliva[,2:3])
```

`createTrees` *Create a Tree Object*

Description

This function creates a list tree objects of type 'phylo' for use in plotting the trees.

Usage

```
createTrees(data, samples = NULL, level = "genus", split = ".")
```

Arguments

`data`	A data frame in which each column contains the rdp read counts for every taxa given in the row names.
`samples`	Deprecated. Only send the columns in data to create.
`level`	The depth the tree creation will go down to (kingdom, phylum, class, order, family, genus, species, subspecies).
`split`	This is the character that separates the taxa levels in the row names.

Details

For 'level' k, p, c, o, f, g, s and ss can be used in place of kingdom, phylum, class, order, family, genus, species and subspecies respectively.

Value

A list of 'phylo' objects that can be passed to plotTree to plot them.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)

### Creates a tree for the 4th sample in 'Saliva'
salivaTree <- createTrees(saliva[,4, drop=FALSE])
```

`displayLegend` *Displays Tree Plot Legend*

Description

This function displays a legend that shows the tree branch sizes/colors divisions.

Usage

```
displayLegend(colors = NULL, divisions = NULL, title = "Confidence Value")
```

Arguments

`colors`	A vector of colors to be used in the plot from lowest ranking to highest ranking.
`divisions`	A vector of numbers from lowest to highest to separate the tree branches into the color ranking.
`title`	The title for the legend.

Details

The values for division should directly relate to the values of your data, i.e. if your data ranges from 0 to 50000 reads you should adjust the divisions to fit your data.

Value

A blank plot that contains a legend.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
displayLegend(c("red", "orange", "blue"), c(.1, 100, 10000))
```

`formatData` *Formats a Data Set*

Description

This function will take a data set and format it by removing low count trees, and/or normalizing counts.

Usage

```
formatData(data, countThreshold = 1000, normalizeThreshold = 10000)
```

Arguments

`data` A data frame in which each column contains the rdp read counts for every taxa given in the row names.

`countThreshold` A cut off threshold for reads - all trees with fewer than this number of reads will be removed.

`normalizeThreshold`
All the trees that are not removed will be normalized to this many reads.

Details

When removing trees with too few reads, the cuts off is based on the value of the top level node, not the sum of all the reads in a sample.

Value

A new data set that is trimmed and standardized based on the specified parameters. The new data is also reordered alphabetically according to row labels.

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Examples

```
data(saliva)

saliva2 <- formatData(saliva, 1000, 10000)
```

`generateTree` *Generate Test Trees*

Description

This function will take several initial trees and will randomly populate new trees based on the originals.

Usage

```
generateTree(data, numReadsPerSamp, theta = NULL, level = "genus", split = ".")
```

Arguments

`data` A data frame in which each column contains the rdp read counts for every taxa given in the row names.

`numReadsPerSamp` A vector specifying the number of reads or sequence depth for each sample.

`theta` When theta is not NULL the base tree is generated by using the `dirmult` function.

`level` The depth the tree will go down to (kingdom, phylum, class, order, family, genus, species, subspecies).

`split` This is the character that separates the taxa levels in the row names.

Value

A data frame containing the generated tree(s).

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Examples

```
data(saliva)

### Generate a the number of reads per sample
### The first number is the number of reads and the second is the number of subjects
nrs <- rep(10000, 2)

gendata <- generateTree(saliva, nrs)
```

---

getMLEandLoglike *Get MLE and Log Likelihood of a Data Set*

Description

This function takes a data set and computes the MLE and its Log-Likelihood value.

Usage

```
getMLEandLoglike(data, maxSteps = 50, weightCols = NULL, delta = 10^(-6), weight = NULL)
```

Arguments

`data`	A data frame in which each column contains the rdp read counts for every taxa given in the row names.
`maxSteps`	The maximum number of times to iterate though for the MLE.
`weightCols`	A vector of weights for the subjects.
`delta`	The minimum threshold of change in f to stop the search for the MLE.
`weight`	Deprecated, use weightCols instead

Details

A unimodal probability model for graph-valued random objects has been derived and applied previously to several types of graphs (cluster trees, digraphs, and classification and regression trees) (For example, Banks and Constantine, 1998; Shannon and Banks, 1999). Here we apply this model to HMP trees constructed from RDP matches. Let G be the finite set of taxonomic trees with elements g, and $d : G \times G \to R^+$ an arbitrary metric of distance on G. We have the probability measure $H(g^*, \tau)$ defined by

$$P(g; g^*, \tau) = c(g^*, \tau) \exp(-\tau d(g^*, g)), forallg \in G,$$

where g^* is the modal or central tree, τ is a concentration parameter, and $c(g^*, \tau)$ is the normalization constant. The distance measure between two trees is the Euclidean norm of the difference between their corresponding adjacency-vectors. To estimate the parameters (g^*, τ), we use the maximum likelihood estimate (MLE) procedure described in La Rosa et al. (see reference 2)

Value

A list containing the MLE, log-likelihood, tau, the number of iterations it took to run, and some intermediate values

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Examples

```
data(saliva)

### We use 1 for the maximum number of steps for computation time
### This value should be much higher to ensure an accurate result
numSteps <- 1

mle <- getMLEandLoglike(saliva, numSteps)$mleTree
```

`mergeDataSets` *Merge Multiple Data Sets*

Description

This function can take any number of data sets, calculate their individual and combined MLEs and then merge them.

Usage

```
mergeDataSets(dataList, calcMLE = FALSE, uniqueNames = FALSE, data = NULL)
```

Arguments

`dataList`	A list of data frames in which each column contains the rdp read counts for every taxa given in the row names.
`calcMLE`	If 'FALSE' the MLEs for the data sets will not be calculated, otherwise they are added to the end.
`uniqueNames`	If 'TRUE' the column names in the combined data set will be appended to insure uniqueness, otherwise the column names will follow the naming process from the `merge` function.
`data`	Deprecated. Replaced with dataList.

Details

Although not required, all data sets should be standardized to the same number of reads before merging.

Value

A single data set containing all the data from the input data sets, in addition to their individual MLEs and a combined MLE if requested.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)
data(stool)

dataComb <- mergeDataSets(list(saliva, stool))
```

---

pairedCompareTwoDataSets

Likelihood-Ratio-Test Statistics to Compare the Distribution of 2 Paired Sets of RDP-Based Taxonomic Trees

---

Description

This functions compares the distribution of two paired sets of RDP-based taxonomic trees using Likelihood-Ratio-Test statistics and a p-value is computed using permutation.

Usage

```
pairedCompareTwoDataSets(data1, data2, numPerms = 1000, parallel = FALSE,
cores = 3, maxSteps=50, delta=10^(-6))
```

Arguments

`data1, data2`	Data frames in which each column contains the rdp read counts for every taxa given in the row names.
`numPerms`	Number of permutations. In practice this should be at least 1,000.
`parallel`	When this is 'TRUE' it allows for parallel calculation of the permutations. Requires the package `doParallel`.
`cores`	The number of parallel processes to run if parallel is 'TRUE'.
`maxSteps`	The maximum number of times to iterate though for the MLE.
`delta`	The minimum threshold of change in f to stop the search for the MLE.

Details

Note: Both data sets should be standardized to the same number of reads.

Value

A p-value for the similarity of the two data sets based on the permutation test.

Author(s)

Patricio S. La Rosa, Elena Deych, Berkley Shands, William D. Shannon

Examples

```
data(saliva)
data(stool)

### We use 1 for the number of permutations for computation time
### This value should be at least 1000 for an accurate result
numPerms <- 1

pval <- pairedCompareTwoDataSets(saliva, stool, numPerms)
pval
```

plotTree *Plots a Tree Object*

Description

This function takes one or more 'phylo' objects and plots them.

Usage

```
plotTree(treeList, colors = NULL, divisions = NULL, main = NULL, sub = "",
showTipLabel = TRUE, showNodeLabel = FALSE, displayLegend = TRUE,
trees = NULL)
```

Arguments

`treeList`	A list that contains at least one tree of type 'phylo'.
`colors`	A vector of colors to be applied to the branches in the plot.
`divisions`	A vector of numbers to be used as break points to assign different colors.
`main`	A custom title(s) for the plot(s).
`sub`	A custom subtitle for the plot.
`showTipLabel`	Hides the tip labels if 'FALSE' otherwise it shows all non-zero tip labels.
`showNodeLabel`	Hides the interior node labels if 'FALSE' otherwise it shows all non-zero node labels.
`displayLegend`	Enables the display of a legend of the branch colors and divisions when 'TRUE'.
`trees`	Deprecated. Replaced with treeList.

Details

Notes:

1. The `phylo` type is a product of the ape package and the `createTrees` function in this package produces a list of `phylo` type objects for use with this function.
2. The values for division should directly relate to the values of your data, i.e. if your data ranges from 0 to 50000 reads you should adjust the divisions to fit your data.

Value

A plot of the tree(s).

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)

### Creates a tree for the 4th sample in 'Saliva' then plots it
salivaTree <- createTrees(saliva[,4, drop=FALSE])
plotTree(salivaTree, displayLegend=FALSE)
```

`plotTreeDataMDS` *Plot an MDS Plot of a Group of Trees*

Description

This function can take any number of data sets and plots them on an MDS plot to show relative closeness to one another.

Usage

```
plotTreeDataMDS(dataList, main = "Tree MDS Comparisons", calcMLE = TRUE,
mleTitles = NULL, dotColors = NULL, dotSizes = NULL, showNames = FALSE,
returnCoords = FALSE, data = NULL)
```

Arguments

`dataList` A list of a data frames in which each column contains the rdp read counts for every taxa given in the row names.

`main` A title for the MDS plot.

`calcMLE` If 'FALSE' the MLEs for the data sets will not be calculated and plotted.

`mleTitles` Deprecated. Replaced with the names in 'dataList'.

`dotColors` The colors to be used when plotting the points and MLE points on the MDS plot.

`dotSizes` A vector in which the first value is the data points CEX and the second value is the MLEs CEX.

`showNames` When 'TRUE' the column name will be plotted above each corresponding point.

`returnCoords` When 'TRUE' this function will return the x and y coordinates for every plotted point.

`data` Deprecated. Replaced with dataList.

Value

A MDS plot of the data.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)
data(stool)

plotTreeDataMDS(list(Saliva=saliva, Stool=stool))
```

saliva *Saliva Data Set*

Description

A data set containing all taxa from 24 subjects.

Usage

```
data(saliva)
```

Format

The format is a data frame of 454 rows by 24 columns, with each column being a separate subject and each row being a different taxa denoted by the row names. The taxanomical levels are separated by a '.' in their names (Bacteria.Phylum.Class....). The values in each column are the sum of values that each taxa had in an RDP file. It should also be noted that the samples are normalized to 7000 reads and any level that ends with a U was unclassified in the RDP file.

stool *Stool Data Set*

Description

A data set containing all taxa from 24 subjects.

Usage

```
data(stool)
```

Format

The format is a data frame of 371 rows by 24 columns, with each column being a separate subject and each row being a different taxa denoted by the row names. The taxanomical levels are separated by a '.' in their names (Bacteria.Phylum.Class....). The values in each column are the sum of values that each taxa had in an RDP file. It should also be noted that the samples are normalized to 7000 reads and any level that ends with a U was unclassified in the RDP file.

throat *Throat Data Set*

Description

A data set containing all taxa from 22 subjects.

Usage

```
data(throat)
```

Format

The format is a data frame of 529 rows by 22 columns, with each column being a separate subject and each row being a different taxa denoted by the row names. The taxanomical levels are separated by a '.' in their names (Bacteria.Phylum.Class....). The values in each column are the sum of values that each taxa had in an RDP file. It should also be noted that the samples have not been normalized and should be used with 'formatData'. Also any level that ends with a U was unclassified in the RDP file.

`trimToTaxaLevel` *Trim a Tree to a Given Level*

Description

This function will take a tree and either remove all nodes lower than the given level or will remove all nodes not of the given level.

Usage

```
trimToTaxaLevel(data, level = "genus", eliminateParentNodes = FALSE,
trimBelow = NULL, split = ".")
```

Arguments

`data` A data frame in which each column contains the rdp read counts for every taxa given in the row names.

`level` The depth the tree will go down to (kingdom, phylum, class, order, family, genus, species, subspecies).

`eliminateParentNodes` If 'TRUE' the data set returned will only contain rows at the level specified by 'myTaxaLevel'. If 'FALSE' the data set returned will contain all the nodes up to the level specified by 'myTaxaLevel'.

`trimBelow` If 'NULL' the function will pull out only the data at the level specified by 'myTaxaLevel'. If 'TRUE' the function will remove all the levels below the specified level. If 'FALSE' the function will remove all the levels above the specified level.

`split` This is the character that separates the taxa levels in the row names.

Details

Notes:

1. For 'level' k, p, c, o, f, g, s and ss can be used in place of kingdom, phylum, class, order, family, genus, species and subspecies respectively.
2. Numbers can also be used for 'level', with no maximum limit.
3. The option to 'eliminateParentNodes' only works when 'trimBelow' is NULL.

Value

A new data set that has been trimmed to the level selected.

Author(s)

Berkley Shands, Patricio S. La Rosa, Elena Deych, William D. Shannon

Examples

```
data(saliva)

### Trims saliva to only contain the class level
salivaClass <- trimToTaxaLevel(saliva, "class", TRUE)
```

Index