

# Package ‘clstutils’

November 20, 2024

**Type** Package

**Title** Tools for performing taxonomic assignment

**Version** 1.54.0

**Depends** R (>= 2.10), clst, rjson, ape

**Imports** lattice, RSQLite

**Suggests** RUnit

**LazyLoad** yes

**LazyData** yes

**Author** Noah Hoffman

**Maintainer** Noah Hoffman <ngh2@uw.edu>

**Description** Tools for performing taxonomic assignment based on phylogeny using pplacer and clst.

**License** GPL-3

**biocViews** Sequencing, Classification, Visualization, QualityControl

**git\_url** <https://git.bioconductor.org/packages/clstutils>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** f4bf8e2

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-20

## Contents

clstutils-package	2
classifyPlacements	2
findOutliers	3
maxDists	4
prettyTree	5
refpkgContents	7
seqdat	8
seqs	9
taxonomyFromRefpkg	9
treeDists	10

<b>Index</b>	<b>13</b>
--------------	-----------

---

clstutils-package      *Sequence based classification and selection of reference sequences.*

---

### Description

Tools for performing taxonomic assignment based on phylogeny using pplacer and clst.

### Details

Package: clstutils  
Type: Package  
Author: Noah Hoffman <ngh2@uw.edu>  
License: GPL3

Index:

### Author(s)

Noah Hoffman

Maintainer: <ngh2@uw.edu>

### See Also

[clst](#)

### Examples

```
library(clstutils)
packageDescription("clstutils")
```

---

classifyPlacements      *Taxonomic classification by phylogenetic placement.*

---

### Description

Given taxonomic information from a reference package and inter-node distances from a reference tree, perform classification of one or more placements provided by pplacer.

### Usage

```
classifyPlacements(taxdata, treedists, placetab, ...,
  verbose = FALSE, debug = FALSE)
```

**Arguments**

taxdata	data.frame, output of <a href="#">taxonomyFromRefpkg</a>
treedists	output of <a href="#">treeDists</a>
placetab	a data.frame with columns at, edge, and branch
...	extra arguments passed to <a href="#">classifyIter</a>
verbose	writes progress messages to terminal if TRUE
debug	be very verbose if TRUE

**Value**

The output is a data.frame describing the taxonomic assignment, along with a description of the confidence of the classification. See the man page for [classify](#) for details on the output.

**Author(s)**

Noah Hoffman

**See Also**

[treeDists](#), [taxonomyFromRefpkg](#)

**Examples**

```
placefile <- system.file('extdata', 'merged.json', package='clstutils')
distfile <- system.file('extdata', 'merged.distmat.bz2', package='clstutils')
refpkgz <- system.file('extdata', 'vaginal_16s.refpkg.tar.gz', package='clstutils')

tmpdir <- tempdir()

orig.dir <- getwd()
setwd(tmpdir)
system(paste("tar --no-same-owner -xzf", refpkgz))
setwd(orig.dir)

refpkg <- file.path(tmpdir, "vaginal_16s.refpkg")

treedists <- treeDists(distfile=distfile, placefile=placefile)
taxdata <- taxonomyFromRefpkg(refpkg, seqnames=rownames(treedists$dmatrix), lowest_rank="species")
placetab <- data.frame(at=49, edge=5.14909e-07, branch=5.14909e-07)
result <- classifyPlacements(taxdata, treedists, placetab)
result
```

---

findOutliers

*Identify outlier objects given a square distance matrix.*

---

**Description**

Outliers are defined as elements with edge length to the centermost element > cutoff. The distance threshold (cutoff) can be either specified, or calculated as a quantile of all pairwise distances in the matrix.

**Usage**

```
findOutliers(mat, quant, cutoff)
```

**Arguments**

mat	square matrix of distances
quant	given all pairwise distances x, calculate distance threshold as quantile(x, quant). Values closer to 0 are more stringent.
cutoff	an absolute cutoff overriding quant

**Value**

Returns a boolean vector corresponding to margin of mat; outliers have a value of TRUE.

**Author(s)**

Noah Hoffman

**Examples**

```
library(ape)
data(seqs)
data(seqdat)
dmat <- ape::dist.dna(seqs[seqdat$tax_name == 'Enterococcus faecium',],
  pairwise.deletion=TRUE, as.matrix=TRUE, model='raw')
summary(dmat[lower.tri(dmat)])
outliers <- findOutliers(dmat, cutoff=0.015)
table(outliers)
```

---

maxDists

*Select a maximally diverse set of items given a distance matrix.*

---

**Description**

Given a square matrix of pairwise distances, return indices of N objects with a maximal sum of pairwise distances.

**Usage**

```
maxDists(mat, idx = NA, N = 1,
  exclude = rep(FALSE, nrow(mat)),
  include.center = TRUE)
```

**Arguments**

mat	square distance matrix
idx	starting indices; if missing, starts with the object with the maximum median distance to all other objects.
N	total number of selections; length of idx is subtracted.
exclude	boolean vector indicating elements to exclude from the calculation.
include.center	includes the "most central" element (ie, the one with the smallest median of pairwise distances to all other elements) if TRUE

**Value**

A vector of indices corresponding to the margin of mat.

**Note**

Note that it is important to evaluate if the candidate sequences contain outliers (for example, mislabeled sequences), because these will assuredly be included in a maximally diverse set of elements!

**Author(s)**

Noah Hoffman

**See Also**

[findOutliers](#)

**Examples**

```
library(ape)
library(clstutils)
data(seqs)
data(seqdat)
efaecium <- seqdat$tax_name == 'Enterococcus faecium'
seqdat <- subset(seqdat, efaecium)
seqs <- seqs[efaecium,]
dmat <- ape::dist.dna(seqs, pairwise.deletion=TRUE, as.matrix=TRUE, model='raw')

## find a maximally diverse set without first identifying outliers
picked <- maxDists(dmat, N=10)
picked
prettyTree(nj(dmat), groups=ifelse(1:nrow(dmat) %in% picked, 'picked', 'not picked'))

## restrict selected elements to non-outliers
outliers <- findOutliers(dmat, cutoff=0.015)
picked <- maxDists(dmat, N=10, exclude=outliers)
picked
prettyTree(nj(dmat), groups=ifelse(1:nrow(dmat) %in% picked, 'picked', 'not picked'),
X = outliers)
```

---

prettyTree

*Draw an annotated phylogenetic tree.*

---

**Description**

Extends [plot.phylo](#) to draw a phylogenetic tree with additional annotation.

**Usage**

```
prettyTree(x, groups, fill,
           X, 0, indices, labels,
           show = rep(TRUE, length(x)),
           largs = list(cex = 0.75, bty = "n", yjust = 0.5),
           parargs = list(mar = c(bottom = 5, left = 2, top = 2,
                                   right = ifelse(is.null(largs), 2, 8))),
           xpd = NA),
           pointargs = list(), glyphs,
           shuffleGlyphs = NA, ...)
```

**Arguments**

x	an object of class phylo, eg <code>x &lt;- nj(ddist)</code>
groups	a factor (or object coercible) to a factor assigning group identity to leaf nodes in x
fill	vector (logical or indices) of points to fill
X	vector of points to mark with an X
0	vector of points to mark with a circle
indices	label points with indices (all points if 'yes', or a subset indicated by a vector)
labels	character vector of tip labels in the same order as <code>x\$tip.label</code>
show	boolean vector of points to show
largs	arguments controlling appearance of the legend or NULL for no legend
parargs	arguments to pass <code>par()</code>
pointargs	arguments to pass <code>points()</code> (other than <code>pch</code> , <code>col</code> , <code>bg</code> )
glyphs	a data.frame with columns named 'col' and 'pch' corresponding to elements of <code>unique(groups)</code>
shuffleGlyphs	NA or an integer (argument to <code>set.seed</code> )
...	passed to <code>plot.phylo</code>

**Details**

`prettyTree` adds to a plot drawn by `plot.phylo`

Vectors specifying annotation should be in the order of row or column labels of the distance matrix used to generate x.

**Value**

Plots to the active device; no return value.

**Note**

See package vignette for additional examples.

**Author(s)**

Noah Hoffman

**See Also**

[plot.phylo](#)

**Examples**

```
library(ape)
data(seqs)
data(seqdat)
prettyTree(nj(dist.dna(seqs)), groups=seqdat$tax_name)
```

---

refpkgContents	<i>Read the contents of a collection of reference sequences ("refpkg").</i>
----------------	---

---

**Description**

Read the manifest file from a repackage and return a list containing the package contents.

**Usage**

```
refpkgContents(path, manifest = "CONTENTS.json")
```

**Arguments**

path	path to a repkg directory
manifest	name of the manifest file

**Value**

Returns a list of lists. Run `example(refpkgContents)` for details.

**Author(s)**

Noah Hoffman

**References**

The description and specification for a reference package can be found in the project repository in github: <https://github.com/fhcrc/taxtastic>

Scripts and tools for creating reference packages are provided in the python package taxonomy, also available from the taxtastic project site.

**See Also**

[taxonomyFromRepkg](#)

**Examples**

```
archive <- 'vaginal_16s.refpkg.tar.gz'
destdir <- tempdir()
system(gettextf('tar -xzf %s --directory="%s"',
               system.file('extdata',archive,package='clstutils'),
               destdir))
refpkg <- file.path(destdir, sub('.tar.gz','',archive))
contents <- refpkgContents(refpkg)
str(refpkg)
```

---

seqdat

*Annotation for the Enterococcus sequence data set.*

---

**Description**

Provides annotation for link{seqs}, an aligned 16S rRNA sequences representing three Enterococcus species.

**Usage**

```
data(seqdat)
```

**Format**

A data frame with 200 observations on the following 5 variables.

seqname a character vector

accession a character vector containing GenBank accession numbers.

tax\_id a character vector

tax\_name a character vector

isType a logical vector indicating if the sequence is from a type strain.

**Source**

These sequences were downloaded from the Ribosomal Database Project website <http://rdp.cme.msu.edu/>

**Examples**

```
data(seqdat)
with(seqdat,{
  table(tax_name, isType)
})
```



---

seqs	<i>Enterococcus sequence data set.</i>
------	--

---

**Description**

Aligned 16S rRNA sequences representing three *Enterococcus* species.

**Usage**

```
data(seqs)
```

**Format**

The format is: 'DNABin' raw [1:200, 1:1848] - - - - ... - attr(\*, "dimnames")=List of 2 ..\$ : chr [1:200] "S000001976" "S000008133" "S000013428" "S000127028" ... ..\$ : NULL

**Source**

These sequences were downloaded from the Ribosomal Database Project website <http://rdp.cme.msu.edu/>

**Examples**

```
data(seqs)
seqs
```

---

taxonomyFromRefpkg	<i>Extract taxonomic information from a refpkg.</i>
--------------------	---

---

**Description**

Construct a data.frame providing the lineage of each sequence represented in the reference package.

**Usage**

```
taxonomyFromRefpkg(path, seqnames, lowest_rank = NA)
```

**Arguments**

path	path to a refpkg directory
seqnames	optional character vector of sequence names. If provided, determines the order of rows in \$taxTab
lowest_rank	name of the most specific (ie, rightmost) rank to include. Default is the name of the rightmost column in refpkg_contents\$taxonomy

**Value**

A list with the following elements:

taxNames	a named character vector of taxonomic names (names are tax_ids)
taxTab	a data.frame in which each row corresponds to a reference sequence and contains a tax_id followed by the corresponding lineage (columns are "root"...lowest_rank)

**Author(s)**

Noah Hoffman

**References**

The description and specification for a reference package can be found in the project repository in github: <https://github.com/fhcrc/taxtastic>

Scripts and tools for creating reference packages are provided in the python package taxonomy, also available from the taxtastic project site.

**See Also**

[refpkgContents](#)

**Examples**

```
archive <- 'vaginal_16s.refpkg.tar.gz'
destdir <- tempdir()
system(gettextf('tar -xzf %s --directory="%s"',
               system.file('extdata',archive,package='clstutils'),
               destdir))
refpkg <- file.path(destdir, sub('.tar.gz','',archive))
reftax <- taxonomyFromRefpkg(refpkg)
str(reftax)
```

---

treeDists

*Provide objects for determining distances among nodes of a reference tree.*

---

**Description**

Provides objects (dists, paths) that can be used to calculate vectors of distances between an internal node and each leaf node. Also returns a square matrix of distances between leaf nodes.

**Usage**

```
treeDists(placefile, distfile)
```

**Arguments**

placefile	path to pplacer output
distfile	path to output of guppy distmat

**Details**

A placement on an edge looks like this:

```
proximal
|
|   d_p
|
|---- x
```

```

|
|   d_d
|
|
|
distal

```

$d_p$  is the distance from the placement  $x$  to the proximal side of the edge, and  $d_d$  the distance to the distal side.

If the distance from  $x$  to a leaf  $y$  is an S-distance  $Q$ , then the path from  $x$  to  $y$  will go through the distal side of the edge and we will need to add  $d_d$  to  $Q$  to get the distance from  $x$  to  $y$ . If the distance from  $x$  to a leaf  $y$  is a P-distance  $Q$ , then the path from  $x$  to  $y$  will go through the proximal side of the edge, and we will need to subtract off  $d_d$  from  $Q$  to get the distance from  $x$  to  $y$ . In either case, we always need to add the length of the pendant edge, which is the second column.

To review, say the values of the two leftmost columns are  $a$  and  $b$  for a given placement  $x$ , and that it is on an edge  $i$ . We are interested in the distance of  $x$  to a leaf  $y$ , which is on edge  $j$ . We look at the distance matrix, entry  $(i,j)$ , and say it is an S-distance  $Q$ . Then our distance is  $Q+a+b$ . If it is a P-distance  $Q$ , then the distance is  $Q-a+b$ .

The distances between leaves should always be P-distances, and there we need no trickery.

(thanks to Erick Matsen for this description)

### Value

A list with the following elements:

<code>dists</code>	rectangular matrix of distances with rows corresponding to all nodes in pplacer order, and columns corresponding to tips in the order of the corresponding phylo{ape} object.
<code>paths</code>	rectangular matrix in the same configuration as <code>dists</code> with values of 1 or -1 if the path between nodes is serial or parallel, respectively (see Details)
<code>dmat</code>	square matrix containing distances between pairs of tips.

### Note

The output of this function is required for `classifyPlacements`.

### Author(s)

Noah Hoffman

### References

Documentation for pplacer and guppy can be found here: <http://matsen.fhcrc.org/pplacer/>

### See Also

[classifyPlacements](#)

**Examples**

```
placefile <- system.file('extdata','merged.json', package='clstutils')
distfile <- system.file('extdata','merged.distmat.bz2', package='clstutils')
treedists <- treeDists(placefile, distfile)

## coordinates of a single placement
placetab <- data.frame(at=49, edge=5.14909e-07, branch=5.14909e-07)

## dvects is a matrix in which each row corresponds to a vector of
## distances between a single placement along the edge of the reference
## tree used to generate 'distfile', and each column corresponds to a
## reference sequence (ie, a terminal node).

dvects <- with(placetab, {
  treedists$dists[at+1,,drop=FALSE] + treedists$paths[at+1,,drop=FALSE]*edge + branch
})
```

# Index

- \* **aplot**
  - prettyTree, 5
- \* **classif**
  - classifyPlacements, 2
  - clstutils-package, 2
  - findOutliers, 3
  - maxDists, 4
  - refpkgContents, 7
  - taxonomyFromRefpkg, 9
  - treeDists, 10
- \* **datasets**
  - seqdat, 8
  - seqs, 9
- \* **package**
  - clstutils-package, 2

classify, 3

classifyIter, 3

classifyPlacements, 2, 11

clst, 2

clstutils (clstutils-package), 2

clstutils-package, 2

findOutliers, 3, 5

maxDists, 4

plot.phylo, 5–7

prettyTree, 5

refpkgContents, 7, 10

seqdat, 8

seqs, 9

taxonomyFromRefpkg, 3, 7, 9

treeDists, 3, 10