Package 'DendroLikeness'

September 6, 2024

Type Package

Title Compare the topology of two dendrograms

Version 1.0

Date 2024-09-05

Author Octavio Martinez

Maintainer <octavio.martinez@cinvestav.mx>

Description Given two dendrograms, you can see which groups are shared between them, obtaining a measure of likeness that is between 0 and 1. Additionally, functions to study one or two lists of sets are provided.

License GPL-3

R topics documented:

`DendroLikeness-package`

Compare the topology of two dendrograms

Description

Given two dendrograms, you can see which groups are shared between them, obtaining a measure of likeness that is between 0 and 1. Additionally, functions to study one or two lists of sets are provided.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
In R you can obtain a matrix of distances between a set of rows using the function `dist()`. Then, a bifurcating dendrogram can be obtained from that distance matrix by using the function `hclust()`. Distinct alternatives exist to estimate the distance matrix, and then different clustering methods can be employed to obtain a bifurcating dendrogram with `hclust()`. This can result in different dendrograms for the same set of rows. To visually detect the differences between any pair of those dendrograms can be complex when the number of rows in the original data is moderately large.

This package gives a two steps solution to the problem of comparing the topology of two dendrograms obtained from the same set of rows. First, function `dend.topo()` gives a description of the topology of a dendrogram and, second, the function `comp.dend()` performs the comparison of the topologies of two dendrograms, summarizing their likeness and giving an explicit list of the clusters that are shared by the two graphs.

Author(s)

Octavio Martinez

Maintainer: <octavio.martinez@cinvestav.mx>

References

Everitt, B. (1974). Cluster Analysis. London: Heinemann Educ. Books.

See Also

`analyze.set`, `analyze2sets`, `dend.topo`, `comp.dend`, `test.set`

Examples

```
# Obtain a dendrogram for 10 objects using a dummy distance matrix
# and the "complete" method:
temp.den1 <- hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5,
dimnames = list(LETTERS[1:10], c(1:5)))), method="complete")
# Optionally you could plot the dendrogram using "plot(temp.den1)"

# Obtain another dendrogram for the same 10 objects but with a different
# distance matrix and method ="average":
temp.den2 <- hclust(dist(matrix(c(c(1:25), c(25:1))/10, nrow = 10,
ncol = 5, dimnames = list(LETTERS[1:10], c(1:5)))), method="average")
# Optionally you could plot the dendrogram using "plot(temp.den2)"

# Obtain the topology of the first dendrogram
temp.to1 <- dend.topo(temp.den1)
# You could examine the original values of the groups in
temp.to1$original
# And also the clusters (groups) formed in
temp.to1$clusters

# Obtain the topology of the second dendrogram
temp.to2 <- dend.topo(temp.den2)
# You could examine the original values of the groups in
temp.to2$original
```

```
# And also the clusters (groups) formed in
temp.to2$clusters

# Now you can compare the two dendrograms with
temp.com1vs2 <- comp.dend(temp.to1, temp.to2)
# This result is a list with components:
names(temp.com1vs2)

# And see some of the components:
temp.com1vs2$basic # The basic results
temp.com1vs2$topo.summary # A summary of topology
temp.com1vs2$identical # Main results

# (see help of the functions for a comprehensive understanding)
# Finally you could remove the temp objects created
rm(temp.den1, temp.den2, temp.to1, temp.to2, temp.com1vs2)
```

`analyze.set` *Analyzes all pairs of components of a list of sets*

Description

If the length of a list of sets is `L`, the number of different pairs sets to be compared will be `L*(L-1)/2`. For each one those comparisons, results are obtained about the number of elements in each set, the number of elements in the union the number of elements in the intersection as well as a description of the comparison.

Usage

```
analyze.set(x = test.set)
```

Arguments

`x`	A `list` of sets that you want to compare by pairs

Value

A `data.frame` in which each row contain a different comparison between two of the sets included in the input list of sets `x`. The output is a `data.frame` with variables: `id.1` - Numerical identifier of the first set to be compared; `id.2` - Numerical identifier of the second set to be compared; `n.1` - Number of elements in set `id.1`; `n.2` - Number of elements in set `id.2`; `n.int` - Number of elements in the intersection of set `id.1` with set `id.2`; `n.uni` - Number of elements in the union of set `id.1` with set `id.2`; `comparison` - A description of the relation between the set `id.1` with set `id.2`. That `comparison` can take values: `disjoint` - when set `id.1` and set `id.2` intersection is empty, or, `identical` - when set `id.1` and set `id.2` are identical, i.e., have the same elements, or, `s1.includes.s2` - when set `id.1` includes all elements in set `id.2`, i.e., set `id.2` is a proper subset of set `id.1`, or, `s2.includes.s1` - when set `id.2` includes all elements in set `id.1`, i.e., set `id.1` is a proper subset of set `id.2`, or, finally, `intersected` - when set `id.1` and set `id.2` have a non empty intersection and do not fulfill any of the above classifications. `s.1` - The explicit set `id.1` and `s.2` - The explicit set `id.2`.

Author(s)

Octavio Martinez

See Also

test.set, analyze2sets

Examples

```
# Make available the default input: a list of 5 sets:
data(test.set)

length(test.set) # How many sets are in the list?
# Define a temporal object from the output of the function
temp.as <- analyze.set(x=test.set)

# Note that the output has 5*(5-1)/2 rows, one for each comparison
nrow(temp.as)

# Let's see the first row which compares the first and second
# elements of test.set,
temp.as[1,]
# Note that even when the two sets, s.1 and s.2 are in different
# order they are the same set.

# Now, let's tabulate the comparisons:
table(temp.as$comparison)

# And see, in turn, each one of the rows with each comparison:
temp.as[temp.as$comparison=="identical",]
temp.as[temp.as$comparison=="disjoint",]
temp.as[temp.as$comparison=="s1.includes.s2",]
temp.as[temp.as$comparison=="s2.includes.s1",]
temp.as[temp.as$comparison=="intersected",]

# Remove the temporal object
rm(temp.as)
```

---

analyze2sets *Analyzes all pairs of components of two lists of sets*

Description

If the length of the first list of sets is L1, and the length of the second list of sets is L2, then the number of pairs of sets to be compared will be L1*L2. For each one those comparisons, results are obtained about the number of elements in each set, the number of elements in the union, the number of elements in the intersection as well as a description of the comparison.

Usage

```
analyze2sets(x = test.set[1:3], y = test.set[2:5], only.identical = FALSE)
```

Arguments

x	The first list of sets that you want to compare by pairs
y	The second list of sets that you want to compare by pairs
only.identical	A logical variable. If TRUE the output will contain only identical sets between the two lists

Value

A `data.frame` in which each row contain a comparison between two of the sets included in the input lists of sets, `x` and `y`. The output `data.frame` has variables: `id.x` - Numerical identifier of the set in `x` to be compared; `id.y` - Numerical identifier of the set in `y` to be compared; `n.x` - Number of elements in set `id.x`; `n.y` - Number of elements in set `id.y`; `n.int` - Number of elements in the intersection of set `id.x` with set `id.x`; `n.uni` - Number of elements in the union of set `id.x` with set `id.x`; `comparison` - A description of the relation between the set `id.x` with set `id.y`. That `comparison` can take values: `disjoint` - when set `id.x` and set `id.y` intersection is empty, or, `identical` - when set `id.x` and set `id.y` are identical, i.e., have the same elements, or, `sx.includes.sy` - when set `id.x` includes all elements in set `id.y`, i.e., set `id.y` is a proper subset of set `id.x`, or, `sy.includes.sx` - when set `id.y` includes all elements in set `id.x`, i.e., set `id.x` is a proper subset of set `id.y`, or, finally, `intersected` - when set `id.x` and set `id.y` have a non empty intersection and do not fulfill any of the above classifications. `s.x` - The explicit set `id.x` and `s.y` - The explicit set `id.y`.

Author(s)

Octavio Martinez

See Also

`test.set`, `analyze.set`

Examples

```
# Make available the default input: a list of 5 sets:
data(test.set)

# Define a temporal object from the output of the function
# (with defaults)
temp.a2s <- analyze2sets(x = test.set[1:3], y = test.set[2:5])

# Given that x has 3 sets while y has 4 sets we must have
# 3*4=12 rows in the result temp.a2s,
nrow(temp.a2s)

# Let's see the first row:
temp.a2s[1,]

# Now, let's tabulate the comparisons:
table(temp.a2s$comparison)

# And see, in turn, each one of the rows with each comparison:
temp.a2s[temp.a2s$comparison=="identical",]
temp.a2s[temp.a2s$comparison=="disjoint",]
temp.a2s[temp.a2s$comparison=="sx.includes.sy",]
temp.a2s[temp.a2s$comparison=="sy.includes.sx",]
temp.a2s[temp.a2s$comparison=="intersected",]

# Also see the output of the function when only
# identical comparisons are requested
# (note that x & y are let at their default values)
analyze2sets(only.identical=TRUE)

# Remove the temporal object
```

```
rm(temp.a2s)
```

---

comp.dend *Compares the topologies of two dendrograms*

Description

Assume that you have two dendrograms obtained from the function `hclust()`. Using the function `dend.topo()` you can obtain the topologies of each one of the two dendrograms. Then, function `comp.dend()` will compare the topologies of the two dendrograms, giving likeness estimates as well as an explicit description of the clusters shared by the two dendrograms. This function is the core of the package.

Usage

```
comp.dend(x = dend.topo(hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5, dimnames = list(LETTER
```

Arguments

`x`	A result of function `dend.topo()` containing the description of the topology of the first dendrogram, say dendrogram `x`
`y`	A result of function `dend.topo()` containing the description of the topology of the second dendrogram, say dendrogram `y`
`compare.groups`	A `logical` variable which by default is `FALSE`. However, if `compare.groups = TRUE`, then the output of `comp.dend()` will include the description of all possible comparisons between pairs of clusters in both dendrograms. In that case function `analyze2sets()` will be used to obtain those comparisons

Value

`basic`	A `numeric` vector with values for variables `n.comp` - The number of comparable clusters in the two dendrograms, `n.ident` - The number of identical clusters in the two dendrograms, `likeness` - Raw likeness between dendrograms, say `n.ident/n.comp` and `w.likeness` - The weighted likeness between dendrograms
`topo.summary`	A `matrix` with 4 rows and columns with the number of clusters of each length (number of elements) that exist in dendrograms `x` or `y`. The names of the rows are: `n.in.x` - The number of clusters (of each length) in the first dendrogram, `n.in.y` - The number of clusters (of each length) in the second dendrogram, `n.ident` - The number of clusters (of each length) which are identical in both dendrograms, and `p.ident` - The proportion of identity (for each length), say, `n.ident` divided by the maximum of `n.in.x` and `n.in.y` in each one of the columns
`identical`	A `data.frame` with information about the clusters that are identical in both dendrograms. The variables in that `data.frame` are: `id` - Numeric identifier of the cluster comparison, `class` - The length (number of elements) in both clusters compared, `id.x` - Numeric identifier of the cluster in the first dendrogram, `id.y` - Numeric identifier of the cluster in the second dendrogram, `height.x` - The height at which cluster `id.x` was formed in the first dendrogram, `height.y` - The height at which cluster `id.y` was formed in the second dendrogram `cluster` - A description of the common cluster in both dendrograms (in set notation and using the labels of the entities)

height.stats
: A matrix containing statistics for the heights of the clusters in both dendrograms. The rows of the matrix are named as All - Corresponding to statistics for all heights, and Identical - Corresponding to statistics only for heights in identical clusters. The columns of the matrix are: n - The number of heights included, Mean.x - Average of heights in the first dendrogram, Mean.y - Average of heights in the second dendrogram, S.x - Standard deviation of heights in the first dendrogram, S.y - Standard deviation of heights in the second dendrogram and p.value - The value of p from the t.test for the means of heights in the first and second dendrograms (not paired test for All and paired test for Identical)

clusters.comparison
: A data.frame which will be present only when the input option compare.groups = TRUE was used. In that case you need to see the help for function analyze2sets() for the content of the data.frame

objects
: A character string with descriptions of the inputs x and y to the function, extracted from the call of the function

call
: The call of the function

Author(s)

Octavio Martinez

See Also

dend.topo, analyze2sets

Examples

```
# Let's obtain an object with the defaults of the function
temp.cd <- comp.dend()

# NOTE: for fully understand the output it will be useful if
# you plot and examine the two dendrograms that are being analyzed.
# dendrogram "x" (in the input of the function) can be plot with
# plot(hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5,
# dimnames = list(LETTERS[1:10], c(1:5))))))
# (to plot x, please input the text after the remarks in the 2 lines above)
# dendrogram "y" (in the input of the function) can be plot with
# plot(hclust(dist(matrix(c(1:50)/9, nrow = 10, ncol = 5,
# dimnames = list(LETTERS[1:10], c(1:5))))))
# (to plot y, please input the text after the remarks in the 2 lines above)

# Now, lets's examine temp.cd
class(temp.cd) # Which class is the object?
names(temp.cd) # Which names has each component?

temp.cd # Examine this object

# Now, obtain a second object but with option "compare.groups=TRUE"
temp.cd2 <- comp.dend(compare.groups=TRUE)
# Note that this object has components:
names(temp.cd2)

# The "clusters.comparison" component is a data.frame with
dim(temp.cd2$clusters.comparison)
```

```
# and variables:
names(temp.cd2$clusters.comparison)
# Let's see the first 6 rows:
head(temp.cd2$clusters.comparison)
# And also table the comparisons
table(temp.cd2$clusters.comparison$comparison)

# Finally remove the temporary objects
rm(temp.cd, temp.cd2)
```

`dend.topo` *Obtains a description of the topology of a dendrogram*

Description

Gives explicit information of all clusters that are present in a dendrogram obtained with the function `hclust()` and also a list with an explicit representation of all clusters as sets of labels

Usage

```
dend.topo(x = hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5, dimnames = list(LETTERS[1:10], c
```

Arguments

`x`	The output of the function `hclust()`

Details

I recommend to enter the raw output of the function `hclust()` (without modifications). Otherwise errors could occur

Value

A list with two components

`original`	A `data.frame` with variables `V1` - First numerical identifier of a group in the original dendrogram, `V2` - Second numerical identifier of a group in the original dendrogram, `clu` - Number of cluster within the dendrogram, `class` - Length (number of elements) of the cluster, `height` - Height at which the cluster was formed.
`clusters`	A `list` in which each element is a character variable with the elements of the corresponding cluster

Note

The elements of the `clusters` `list` are sets of the original labels in the dendrogram

Author(s)

Octavio Martinez

See Also

`comp.dend`

Examples

```
# Create a temporal object with the output of the function
temp.dt <- dend.topo()

# Note that the default of the function is
# x = hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5,
#     dimnames = list(LETTERS[1:10], c(1:5)))))
# i.e., a dendrogram obtained from the Euclidean distance of an
# arbitrary matrix. Optionally you could plot such dendrogram
# for a better understanding:
# plot(hclust(dist(matrix(c(1:50)/10, nrow = 10, ncol = 5,
# dimnames = list(LETTERS[1:10], c(1:5))))))

# Note that
class(temp.dt)

# With components
names(temp.dt)

# Now, the first component is a data.frame
temp.dt$original

# And the second a list with all the clusters
temp.dt$clusters

# You could analyze that list of clusters
# for example with
# analize.set(temp.dt$clusters)
# which gives a data.frame with 36 rows.
# See the last two rows with
# tail(analize.set(temp.dt$clusters), 2)

# Finally remove the temporal object
rm(temp.dt)
```

---

test.set *A list of 5 small sets*

Description

The list is formed by five small sets of capital letters

Usage

```
data("test.set")
```

Format

The format is: List of 5 $: chr [1:3] "A" "B" "C" $: chr [1:3] "B" "A" "C" $: chr [1:2] "D" "C" $: chr [1:3] "B" "D" "C" $: chr "D"

Examples

```
data(test.set)

# Obtain the union of the five sets
union(test.set[[1]], union(test.set[[2]],
union(test.set[[3]], union(test.set[[4]],
test.set[[5]])))

# The union of sets 1 and 3 is equivalent to previous command
union(test.set[[1]], test.set[[3]])

# A non-empty intersection
intersect(test.set[[3]], test.set[[4]])

# The intersection of the five sets (an empty set)
intersect(test.set[[1]], intersect(test.set[[2]],
intersect(test.set[[3]], intersect(test.set[[4]],
test.set[[5]])))
```

Index