Guide for the use of the "DendroLikeness" package

OCTAVIO MARTÍNEZ
COMPUTATIONAL BIOLOGY LABORATORY, CINVESTAV - IRAPUATO, MÉXICO.

1. Introduction

A "*Dendrogram*" is a diagram resembling a tree and representing relations between entities, as for example taxonomical units (species) or others of interest. Here we are interested in bifurcating dendrograms as the ones produced by the "`hclust()`" R function, which performs "*hierarchical clustering*" (Everitt et al., 2011) from a matrix of distances between entities. For a very good summary of the standard agglomerative hierarchical clustering methods see Table 4.1 in p. 79 of Everitt et al. (2011).

There are two main aspects in a dendrogram, say, its "*topology*" —that is, the description of how the entities form clusters in a hierarchical way— and the heights at which such clusters are formed. Both aspects depend of the method of clustering used by the "`hclust()`" function from a particular matrix of distances.

For the same set of entities and using the same data, different dendrograms can be obtained by varying the distance definition, i.e., the `method` in the `dist()` function, but also by varying the clustering algorithm in the `hclust()` function. Then the problem is to assess the likeness between two or more dendrograms. The functions in the "**DendroLikeness**" R package help with that problem, which has been studied in multiple articles; see for example (Vidovic, 2019; Morlini and Zani, 2012).

Here it is assumed that the reader is familiar with the R environment for statistical computing (R Core Team, 2013); if that is not your case it will be difficult to follow and understand the material. Previously you must have installed the "**DendroLikeness**" R package from the file

"`DendroLikeness_1.0.tar.gz`".

It is also assumed that you have at hand the manual of the package (file "`DendroLikeness-manual.pdf`"), which complements the on-line help for the package.

R input and output will be presented within text boxes, and it is recommended that you replicate the calculations in those boxes, and, even better try variations of the input presented. In the boxes it will be obvious in which cases omit the R prompt, "`>`"; i.e., you can copy the orders given **after** the prompt and past such line in your command R window to replicate the output. In many cases there are lines beginning with the remark symbol, that in R is "`#`". Of course you do not need to copy the remarks in the commands window; they are there to guide you about what is going on. If you are unfamiliar with any function given in a box, the first thing to do is to consult the R help for that function for example, if you see "`dist()`" in a box, you may use "`> ?  dist()`" to invoque the help about that function.

2. Creating some dendrograms to be examined

The number of potentially different bifurcating dendrograms grows very fast as function of the number of entities studied. If you have n entities, the number of different topologies for bifurcating trees (for $n > 2$) is given by

$$d = \frac{(2n-3)!}{2^{(n-2)}(n-2)!}$$

Date: September 9, 2024.

(Felsenstein, 1978) Thus, for example with $n = 15$ we have that $d > 2.1 \times 10^{14}$ –a very large number indeed, and this without taking into account that even when two dendrograms could present the same topology (form) the heights at which the clusters are formed could differ.

We need to create some dendrograms and `Box 1` shows the procedure to do so in R.

```
--------------------------------------------------------------------------------
# Box 1. Estimating some examples of dendrograms.

# FIRST of all call the package:
> library(DendroLikeness)
# (it is assumed that you will done that in all boxes of this document)

### Extra: Felsestein's formula for number of topologies (not needed below)
> felsestein.for <- function(n){factorial(2*n-3)/((2^(n-2))*factorial(n-2))}
# Some evaluations:
> felsestein.for(3)
[1] 3
> felsestein.for(4)
[1] 15
> felsestein.for(15)
[1] 2.13458e+14
# End's Extra

# Obtaining a matrix of random numbers to operate
> set.seed(1959) # For you to obtain the same results that are presented here.
> temp.m <- matrix(runif(225), nrow=15, ncol=15, dimnames=list(LETTERS[1:15], c(1:15)))

# Just for curiosity, see some of the elements of that matrix:
> temp.m[1:5, 1:5] # First 5 rows and columns
           1          2          3          4          5
A 0.02393035 0.45598718 0.51671186 0.08710759 0.92828379
B 0.24120186 0.04074268 0.96773984 0.32128156 0.82687560
C 0.90146718 0.19134374 0.57474474 0.84950395 0.06480590
D 0.60860895 0.76121089 0.99095111 0.85793874 0.38811437
E 0.76064566 0.43049376 0.01818602 0.17498517 0.02543415

# We are going to obtain the Euclidean distances between all the
# 15*(15-1)/2 = 105 different pairs of rows of the matrix.
# Obtain a matrix of distances between rows (units, labeled by letters)
> temp.d <- dist(temp.m, method = "euclidean")

# See the attributes of that object (not shown here)
> attributes(temp.d)

# See some of the elements of temp.d as matrix
> dim(as.matrix(temp.d))
[1] 15 15
> as.matrix(temp.d)[1:5, 1:5]
         A        B        C        D        E
A 0.000000 1.713178 2.353588 1.939625 1.845089
B 1.713178 0.000000 1.823295 1.604697 1.821558
C 2.353588 1.823295 0.000000 1.425934 1.508235
```

```
D 1.939625 1.604697 1.425934 0.000000 1.721341
E 1.845089 1.821558 1.508235 1.721341 0.000000

# List the methods of the hclust function that we will to use
# Input the following:
temp.met <- c("ward.D", "ward.D2", "single", "complete", "average", "mcquitty")
# Now you have
> temp.met
[1] "ward.D"   "ward.D2"  "single"   "complete" "average"
[6] "mcquitty"

# Now we will use those 6 methods to obtain dendrograms.
# List to contain the hclust results
> temp.hc.list <- vector("list", 6)
> names(temp.hc.list) <- temp.met
# Fill that list.
for(i in 1:6){
temp.hc.list[[i]] <- hclust(d = temp.d, method = temp.met[i])
}

# Let's see some of the components of the first dendrogram
> names(temp.hc.list)[1] # The method used:
[1] "ward.D"
> names(temp.hc.list[[1]]) # Components of that dendrogram
[1] "merge"       "height"      "order"       "labels"
[5] "method"      "call"        "dist.method"
> head(temp.hc.list[[1]]$merge) # Which elements merge..
     [,1] [,2]
[1,]   -2  -12
[2,]   -7   -8
[3,]  -14  -15
[4,]   -4   -9
[5,]   -1   -6
[6,]  -10  -11
> temp.hc.list[[1]]$height # at which height each cluster is obtained...
 [1] 1.054120 1.114540 1.207564 1.288619 1.288874 1.300927
 [7] 1.373012 1.374996 1.633775 1.729584 1.761917 2.246137
[13] 2.394811 2.677304
> temp.hc.list[[1]]$labels # Names of the labels
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O"
> temp.hc.list[[1]]$method
[1] "ward.D"
> temp.hc.list[[1]]$call
hclust(d = temp.d, method = temp.met[i])
> temp.hc.list[[1]]$dist.method
[1] "euclidean"
# NOW, let's plot all the six dendrograms:
for(i in 1:6){
plot(temp.hc.list[[i]], main=paste("Method =", temp.met[i]))
readline(prompt = "Hit return to see the next plot")
}
# The results are in Figure 1 and Figure 2 in the text.
```

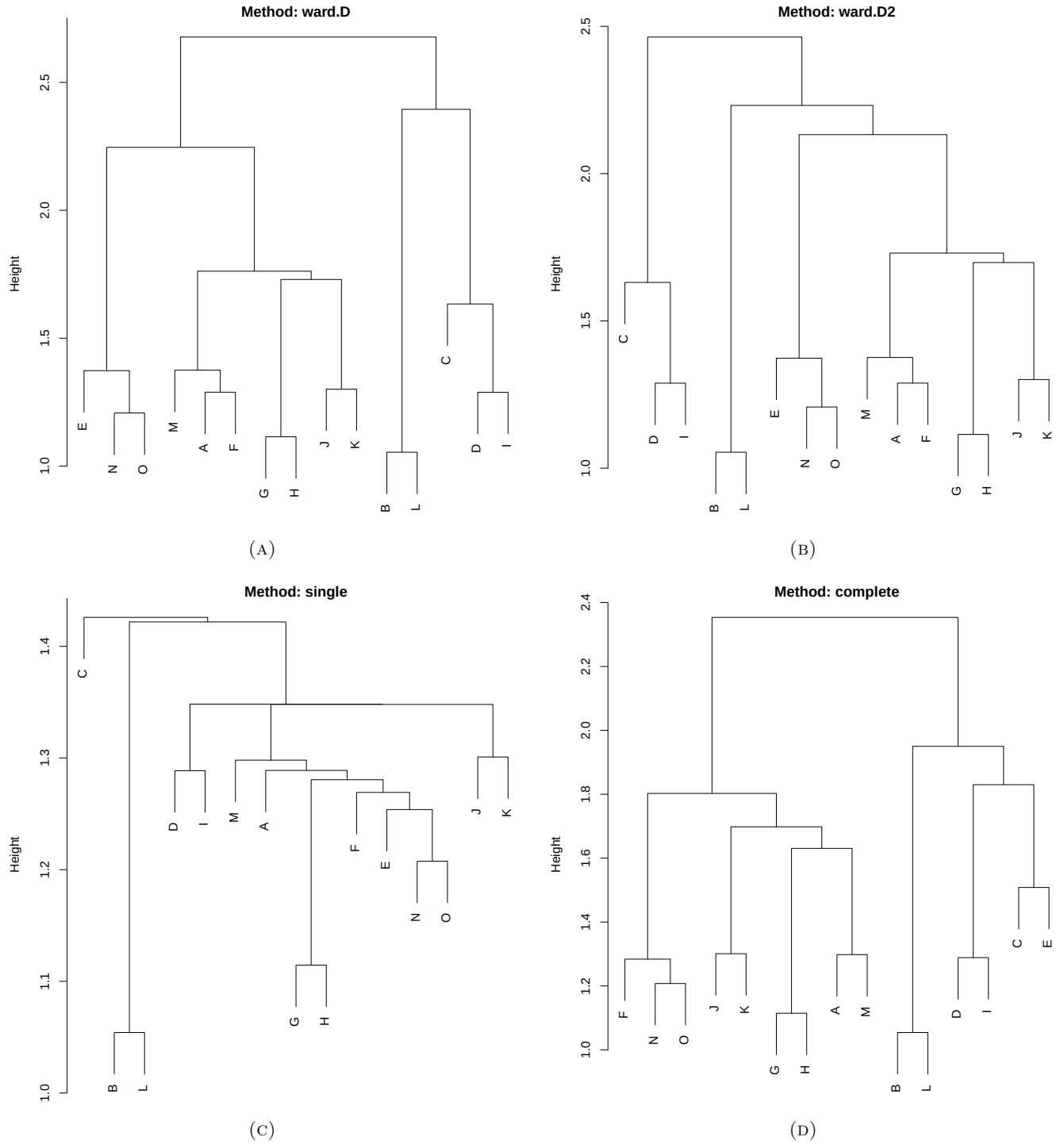
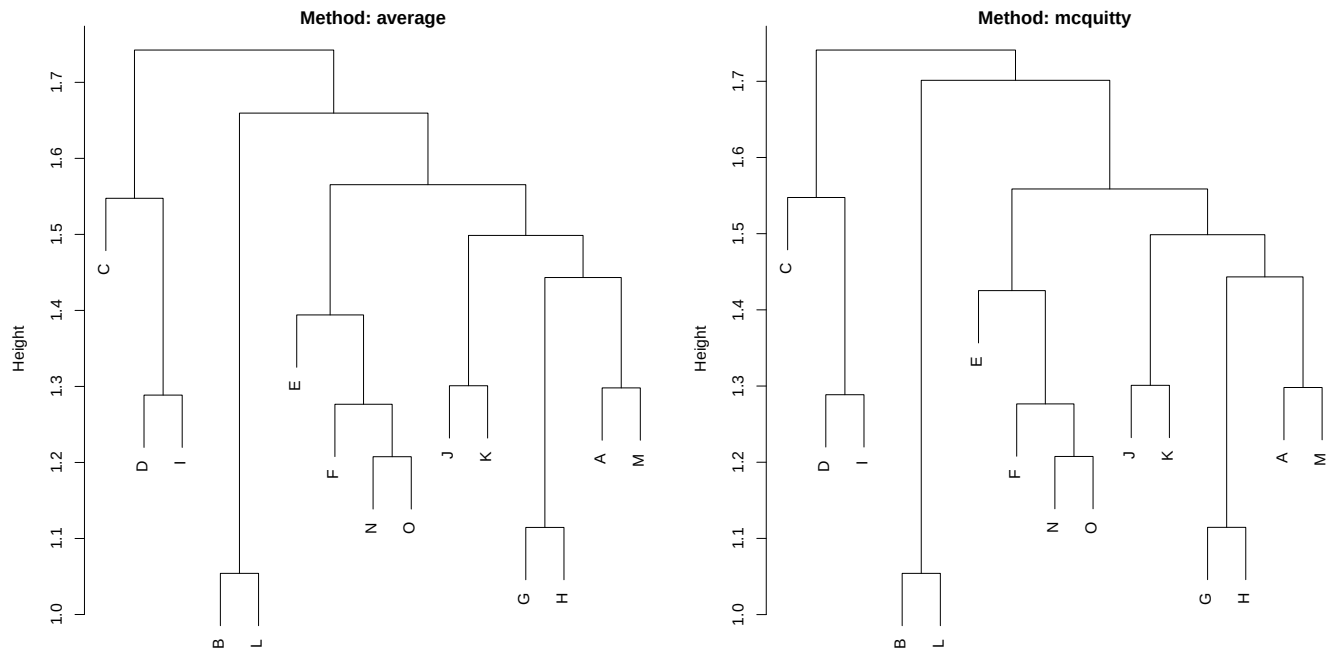


FIGURE 1. Dendrograms obtained for 15 entities from the same distance matrix using different methods.

Above we mentioned that with $n = 15$ entities (letters in `Box 1`) we could obtain many different dendrograms. Figures 1 and 2 present the 6 dendrograms obtained in `Box 1` with the same distance matrix but varying the method of the `hclust()` function to be "`ward.D`", "`ward.D2`", "`single`", "`complete`", "`average`" and "`mcquitty`", respectively.



(A) (B)

FIGURE 2. Dendrograms obtained for 15 entities from the same distance matrix using different methods (continues from Figure 1).

How different are the dendrograms in Figures 1 and 2? First note that all possible bifurcating dendrograms formed for n entities will have $n-1$ clusters; in our case $n = 15$, thus each one of the 6 dendrograms shown has 14 clusters (this understanding that the "root" of the dendrogram, shown at the top of the graphs will include all the 15 entities, and thus that cluster is identical in all 6 dendrograms). This question is also complicated by the fact that the same dendrogram for n entities can be drawn in 2^{n-2} different ways, by rotating each one of the $n-2$ nodes that denote the clusters.

Let's focus our attention in the dendrograms obtained by methods "`ward.D`" and "`ward.D2`" –panels (A) and (B) in Figure 1, respectively. At first glance those two dendrograms are "very alike", but it is difficult to pinpoint exactly where the differences reside. For example, at the right hand side of dendrogram (A) obtained with method "`ward.D`" we find the cluster {C, D, I} that then groups with {B, L} to form cluster {B, L, C, D, I}. In contrast, the cluster {B, L, C, D, I} is not in the dendrogram (B) obtained with method "`ward.D2`"; there is one difference between the topologies of the dendrograms in panels (A) and (B) in Figure 1.

The process of finding the differences in topologies between two dendrograms by looking at the graphs is inefficient and prone to error, but in contrast it could be easily performed by an algorithm, as the one implemented in the function "`comp.dend()`" in our package.

`Box 2` presents the R code to obtain the topologies of the dendrograms in Figures 1 and 2 as well as to compare all pairs of topologies of those dendrograms.

```
-------------------------------------------------------------------------------------
# Box 2. Obtaining and comparing the topologies of the 6 dendrograms.

# Previously, in Box 1, we obtained a list with the 6 dendrograms in object "temp.hc.list"
# (we also have the methods employed in "temp.met").

# Obtain a list with the topologies of each one of the 6 dendrograms using "dend.topo"
```

```
# Define a list for results of dend.topo()
> temp.dend.topo <- vector("list", 6)
> names(temp.dend.topo) <- temp.met

# Fill that list
for(i in 1:6){
temp.dend.topo[[i]] <- dend.topo(temp.hc.list[[i]])
}

# Now we are ready to compare all pairs of topologies of the 6 dendrograms.
# Because we have 6 dendrograms, the number of comparisons that need to be
# performed are:
> 6*(6-1)/2
[1] 15

# Thus we can prepare a data.frame to keep the 15 results of comparison pairs:
> temp.comp <- data.frame(id.com=c(1:15), id.den1=NA, id.den2=NA, met.den1="", met.den2="")

# Fill that data.frame
k <- 0
for(i in 1:5){
for(j in (i+1):6){
k <- k+1
temp.comp$id.den1[k] <- i
temp.comp$id.den2[k] <- j
temp.comp$met.den1[k] <- temp.met[i]
temp.comp$met.den2[k] <- temp.met[j]
}
}

# Thus we obtain:
> temp.comp
   id.com id.den1 id.den2 met.den1 met.den2
1       1       1       2   ward.D  ward.D2
2       2       1       3   ward.D   single
3       3       1       4   ward.D complete
4       4       1       5   ward.D  average
5       5       1       6   ward.D mcquitty
6       6       2       3  ward.D2   single
7       7       2       4  ward.D2 complete
8       8       2       5  ward.D2  average
9       9       2       6  ward.D2 mcquitty
10     10       3       4   single complete
11     11       3       5   single  average
12     12       3       6   single mcquitty
13     13       4       5 complete  average
14     14       4       6 complete mcquitty
15     15       5       6  average mcquitty

# Let's make a list with the results of comparisons
# using the function "comp.dend()"
> temp.comp.dend <- vector("list", 15)
```

```
> names(temp.comp.dend) <- paste(temp.comp$met.den1, temp.comp$met.den2, sep="-VS-")
> head(names(temp.comp.dend))
[1] "ward.D-VS-ward.D2"  "ward.D-VS-single"   "ward.D-VS-complete"
[4] "ward.D-VS-average"  "ward.D-VS-mcquitty" "ward.D2-VS-single"

for(i in 1:15){
temp.comp.dend[[i]] <- comp.dend(temp.dend.topo[[temp.comp$id.den1[i]]],
     temp.dend.topo[[temp.comp$id.den2[i]]])
}

# Each one of the 15 comparisons is a list with seven components, for example
> names(temp.comp.dend[[1]])
[1] "basic"        "topo.summary" "identical"    "height.stats"
[5] "objects"      "call"

# Later we will carefully examine some of the results, but here note that
# the basic results (for each comparison) are in component "basic", for example:
> temp.comp.dend[[1]]$basic
    n.comp    n.ident   likeness w.likeness
13.0000000 12.0000000  0.9230769  0.7142857
> temp.comp.dend[[14]]$basic # For comparison 14
    n.comp    n.ident   likeness w.likeness
13.0000000  9.0000000  0.6923077  0.3367347

# Note that for all 15 comparisons, the number of comparable
# sets is the same: 13

# Now, let's complement the information in data.frame "temp.comp" with
# variables:
> temp.comp$n.ident <- NA
> temp.comp$likeness <- NA
> temp.comp$w.likeness <- NA

# And let's fill those variables
for(i in 1:15){
temp.comp[i, 6:8] <- temp.comp.dend[[i]]$basic[2:4]
}

# Make a summary of those variables:
> summary(temp.comp[, 6:8])
    n.ident          likeness        w.likeness
 Min.   : 5.000   Min.   :0.3846   Min.   :0.07143
 1st Qu.: 7.000   1st Qu.:0.5385   1st Qu.:0.22870
 Median : 7.000   Median :0.5385   Median :0.27083
 Mean   : 7.733   Mean   :0.5949   Mean   :0.33734
 3rd Qu.: 8.500   3rd Qu.:0.6538   3rd Qu.:0.39456
 Max.   :13.000   Max.   :1.0000   Max.   :1.00000

# And also see the Pearson's correlations between them:
> cor(temp.comp[,6:8])
             n.ident  likeness w.likeness
n.ident    1.0000000 1.0000000  0.9558736
```

```
likeness   1.0000000 1.0000000  0.9558736
w.likeness 0.9558736 0.9558736  1.0000000

# Let's also tabulate the values of temp.comp$n.ident
> table(temp.comp$n.ident)
 5  7  8  9 12 13
 3  6  2  2  1  1

# And let's see the temp.comp data.frame ordered by the weighted likeness
# (only more relevant variables)
> temp.comp[order(temp.comp$w.likeness), 4:8]
   met.den1 met.den2 n.ident  likeness w.likeness
10   single complete       5 0.3846154 0.07142857
3    ward.D complete       5 0.3846154 0.08928571
7   ward.D2 complete       5 0.3846154 0.08928571
2    ward.D   single       7 0.5384615 0.21666667
6   ward.D2   single       7 0.5384615 0.24074074
11   single  average       7 0.5384615 0.25925926
12   single mcquitty       7 0.5384615 0.25925926
4    ward.D  average       7 0.5384615 0.27083333
5    ward.D mcquitty       7 0.5384615 0.27083333
13 complete  average       9 0.6923077 0.33673469
14 complete mcquitty       9 0.6923077 0.33673469
8   ward.D2  average       8 0.6153846 0.45238095
9   ward.D2 mcquitty       8 0.6153846 0.45238095
1    ward.D  ward.D2      12 0.9230769 0.71428571
15  average mcquitty      13 1.0000000 1.00000000

# We will focus in understanding the following comparisons:
> temp.comp[c(10,3,7,2),c(1,4:8)]
   id.com met.den1 met.den2 n.ident  likeness w.likeness
10     10   single complete       5 0.3846154 0.07142857
3       3   ward.D complete       5 0.3846154 0.08928571
7       7  ward.D2 complete       5 0.3846154 0.08928571
2       2   ward.D   single       7 0.5384615 0.21666667
# (relevant dendrograms are presented in Figure 3).
```

In `Box 2` we obtained a list with the 6 dendrogram topologies (object "`temp.dend.topo`") and also a list with all 15 posible comparisons by pairs of the topologies of the dendrograms (object "`temp.comp.dend`") and we have a summary of the differences in the object "`temp.comp`".

In Figure 3 we have the plot with four of the dendrograms, and in the last table in `Box 2` we see a summary of 4 of the comparisons among those dendrograms. We will focus and understand how the likeness coefficients between dendrograms, say the raw likeness (variable `likeness`) and mainly the weighted likeness (variable `w.likeness`) are computed.

When comparing the methods "single" versus "complete", see panels (A) and (B) in Figure 3, the values for likeness are `likeness = 0.3846154` and `w.likeness = 0.07142857`. Let's see that in that comparison we have

```
> names(temp.comp.dend)[10] # The comparison of interest
[1] "single-VS-complete"
> names(temp.comp.dend[[10]]) # Names of components
```

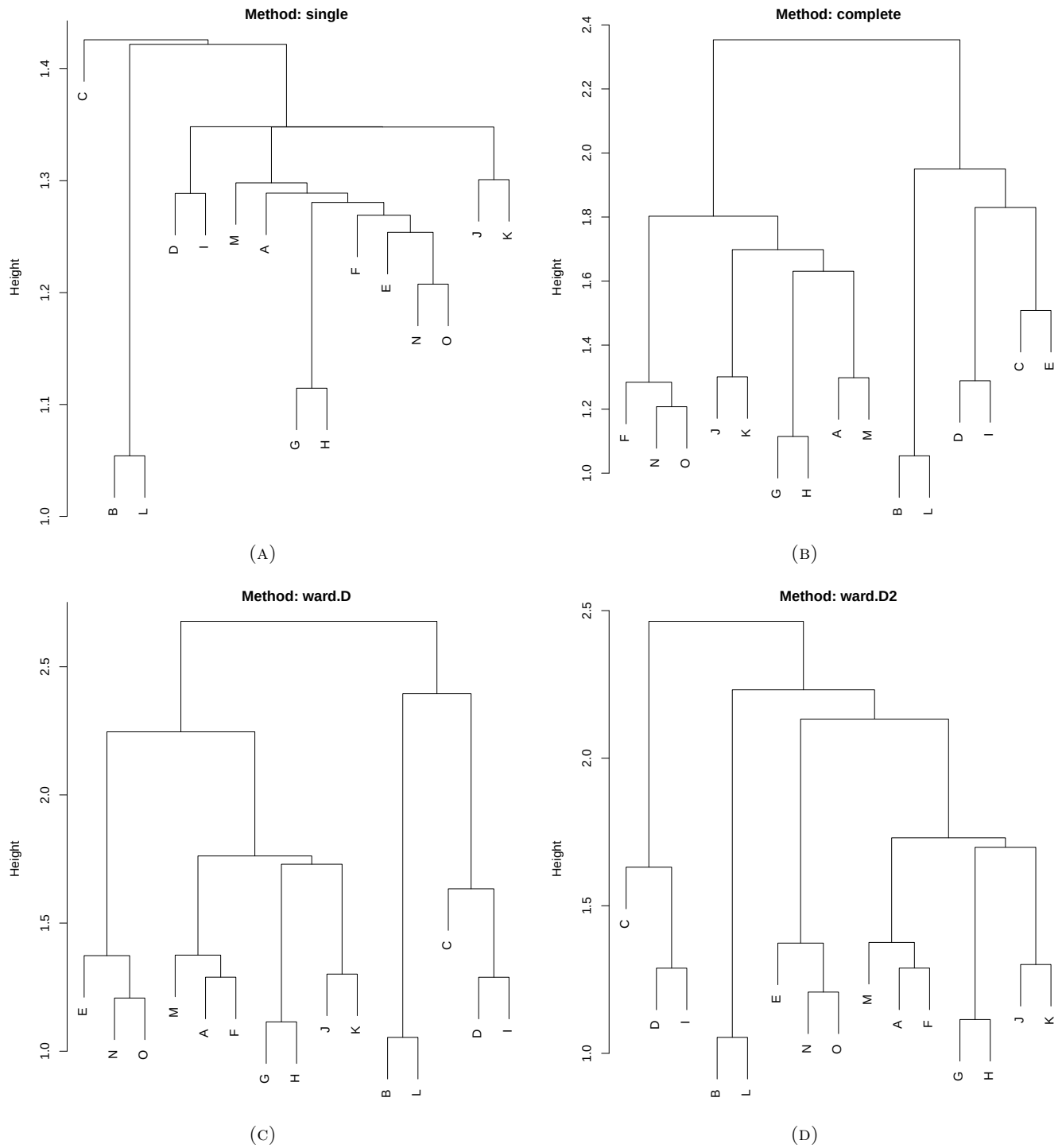


FIGURE 3. Dendrograms with methods "single" (A), "complete" (B), "ward.D" (C) and "ward.D2" (D).

```
[1] "basic"        "topo.summary" "identical"    "height.stats"
[5] "objects"      "call"
# And let's see
> temp.comp.dend[[10]]$basic
     n.comp     n.ident    likeness  w.likeness
13.00000000  5.00000000  0.38461538  0.07142857
```

```
> temp.comp.dend[[10]]$topo.summary
                2 3 4 6 7 8 9 10 12 14
n.in.x  5.0000000 1 1 1 1 1 0  1  1  1
n.in.y  7.0000000 1 2 2 0 0 1  0  0  0
n.ident 5.0000000 0 0 0 0 0 0  0  0  0
p.ident 0.7142857 0 0 0 0 0 0  0  0  0
> mean(temp.comp.dend[[10]]$topo.summary[4,])
[1] 0.07142857
> temp.comp.dend[[10]]$identical$cluster
[1] "{B, L}" "{G, H}" "{N, O}" "{D, I}" "{J, K}"
```

Between methods "single" and "complete" we have 5 of 13 identical clusters, which are "{B, L}" "{G, H}" "{N, O}" "{D, I}" and "{J, K}", thus the raw likeness is $5/13 \approx 0.3846$; however, in `topo.summary` we see that there are clusters with number of elements 2, 3, 4, 6, 7, 8, 9, 10, 12 and 14 in either the `x` or `y` dendrograms, nevertheless only in the class with two elements there are 5 shared clusters giving a proportion of identities (`p.ident`) equal to 5/7=0.7142857 but for all the other 9 categories of clusters with 3, 4, 6, 7, 8, 9, 10, 12 or 14 elements there are not shared clusters and thus the values of `p.ident` are zero. The weighted likeness between dendrograms, "`w.likeness`", is defined as the average of the proportions of identities, `p.ident`, and in that case the value of that coefficient is ≈ 0.0714

The relevance of the weighted likeness between dendrograms, "`w.likeness`", is that it can discriminate more accurately some cases where the raw likeness, "`likeness`", is identical. For example, in both "single" versus "complete" and "ward.D" versus "complete" we have 5/13 common clusters, thus the values of "`likeness`" are identical, however note that in "ward.D" versus "complete" we have

```
> names(temp.comp.dend)[3] # The comparison of interest
[1] "ward.D-VS-complete"
> temp.comp.dend[[3]]$basic
     n.comp     n.ident    likeness  w.likeness
13.00000000  5.00000000  0.38461538  0.08928571
> temp.comp.dend[[3]]$topo.summary
> temp.comp.dend[[3]]$topo.summary
                2 3 4 5 6 7 9 10
n.in.x  6.0000000 3 1 1 0 1 0  1
n.in.y  7.0000000 1 2 0 2 0 1  0
n.ident 5.0000000 0 0 0 0 0 0  0
p.ident 0.7142857 0 0 0 0 0 0  0
> mean(temp.comp.dend[[3]]$topo.summary[4,])
[1] 0.08928571
> temp.comp.dend[[3]]$identical$cluster
[1] "{B, L}" "{G, H}" "{N, O}" "{D, I}" "{J, K}"
```

thus, even when the identical clusters are the same the number of classes of comparable clusters is only eight, say 2, 3, 4, 5, 6, 7, 9 and 10, and then the value of "`w.likeness`" is ≈ 0.0893

In summary, the function `comp.dend()` allows an efficient way to assess the likeness between dendrograms.

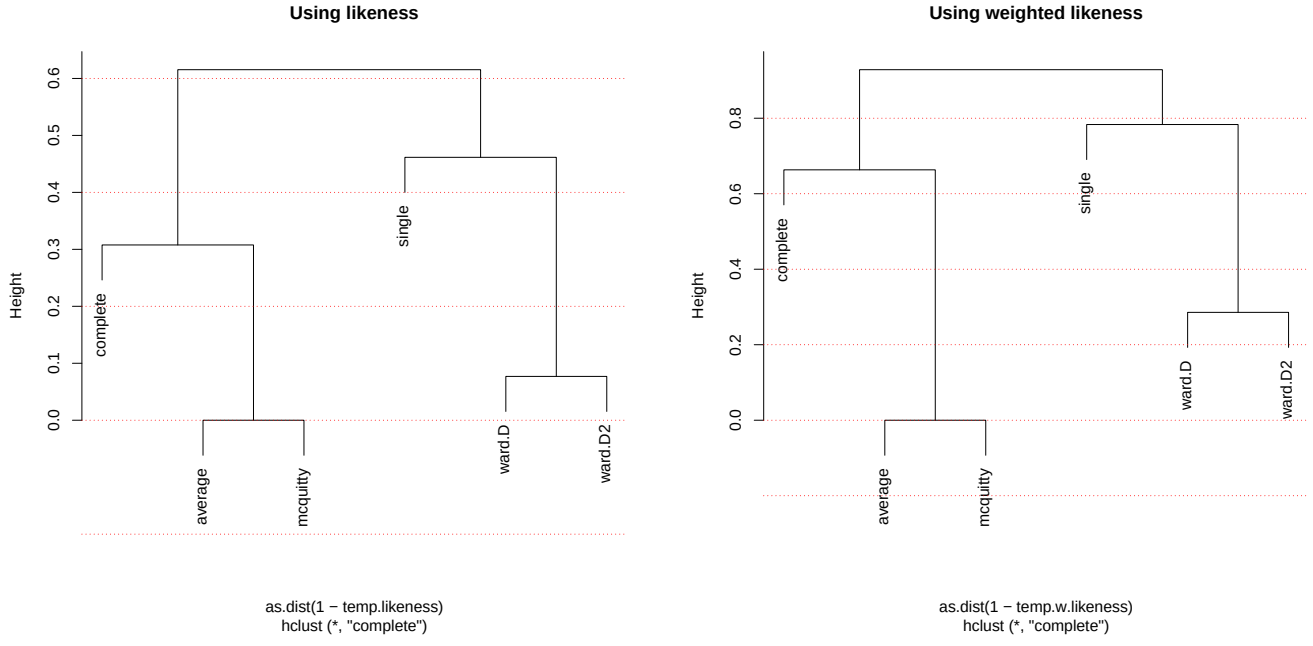
3. Obtaining a Dendrogram of Dendrograms

From the same artificial data, and using 6 different clustering methods we obtained 6 dendrograms. Here we want to summarize in some way how different are those 6 dendrograms. With that aim we can employ the measures `likeness` or `w.likeness` that we have in the object `temp.comp`, which has the $6\times(6-1)/2 = 15$ different comparisons between dendrograms. We can re-order `likeness` and `w.likeness`

in a matrix and from that matrix obtain the distances between dendrograms, keeping in mind that both, `likeness` and `w.likeness` are coefficients that vary between 0 and 1. Then, having obtained all pairs of distances between dendrograms we can obtain a dendrogram showing how the 6 dendrograms clusters, and thus giving a summary of the differences found between methods. `Box 3` presents the R calculations while Figure 4 show the dendrograms obtained from 1-`likeness` and 1-`w.likeness` in panels (A) and (B), respectively.

```
# Box 3: Dendrograms of dendrograms.
# Previously we obtained the values of likeness and weighted likeness between
# all 15 pairs of dendrograms, see:
> nrow(temp.comp)
[1] 15
> head(temp.comp, 2) # The first two rows
  id.com id.den1 id.den2 met.den1 met.den2 n.ident  likeness
1      1       1       2   ward.D  ward.D2      12 0.9230769
2      2       1       3   ward.D   single       7 0.5384615
  w.likeness
1  0.7142857
2  0.2166667

# Now the idea is to compare the dendrograms using as distances the values
# of 1 - likeness and 1 - w.likeness. For this we create:

# A temporary matrix as template
temp.x <- matrix(1, nrow=6, ncol=6, dimnames=list(temp.met, temp.met))

# A matrix of likeness between dendrograms:
temp.likeness <- temp.x

# And fill all the values in temp.likeness
k <- 0
for(i in 1:5){
for(j in (i+1):6){
k <- k+1
temp.likeness[i, j] <- temp.comp$likeness[k]
temp.likeness[j, i] <- temp.likeness[i, j]
}
}

# See the first 2 rows:
> temp.likeness[1:2,]
           ward.D   ward.D2    single  complete   average
ward.D  1.0000000 0.9230769 0.5384615 0.3846154 0.5384615
ward.D2 0.9230769 1.0000000 0.5384615 0.3846154 0.6153846
         mcquitty
ward.D  0.5384615
ward.D2 0.6153846

# A matrix of w.likeness between dendrograms:
temp.w.likeness <- temp.x
k <- 0
```

```
for(i in 1:5){
for(j in (i+1):6){
k <- k+1
temp.w.likeness[i, j] <- temp.comp$w.likeness[k]
temp.w.likeness[j, i] <- temp.w.likeness[i, j]
}
}

# See the first 2 rows:
> temp.w.likeness[1:2,]
           ward.D   ward.D2    single   complete   average
ward.D  1.0000000 0.7142857 0.2166667 0.08928571 0.2708333
ward.D2 0.7142857 1.0000000 0.2407407 0.08928571 0.4523810
         mcquitty
ward.D  0.2708333
ward.D2 0.4523810

### Obtaining dendrograms of dendrograms with the distances
# defined as 1 - likeness or 1 - w.likeness, respectively
# (in both cases using the method "complete" that is the default)

# Figure 4 (A)
> plot(hclust(as.dist(1-temp.likeness)), main="Using likeness")

# Figure 4 (B)
> plot(hclust(as.dist(1-temp.w.likeness)), main="Using weighted likeness")
```



(A) (B)

FIGURE 4. A dendrogram of previously obtained dendrograms (see `Box 3`).

In Figure 4 we can see that both of the dendrograms using 1-`likeness` –panel (A), or 1-`w.likeness` –panel (B), have exactly the same topology, however they differ in the height at which the groups are formed. In both cases methods "average" and "mcquilty" are grouped at a distance of zero, and that is because the corresponding dendrograms are identical. Nevertheless for all the other groups the height at which the groups are obtained is larger for 1-`w.likeness` (B) than for 1-`likeness` (A), and that is because `w.likeness` < `likeness` in all cases, given that `w.likeness` has a larger discriminating power.

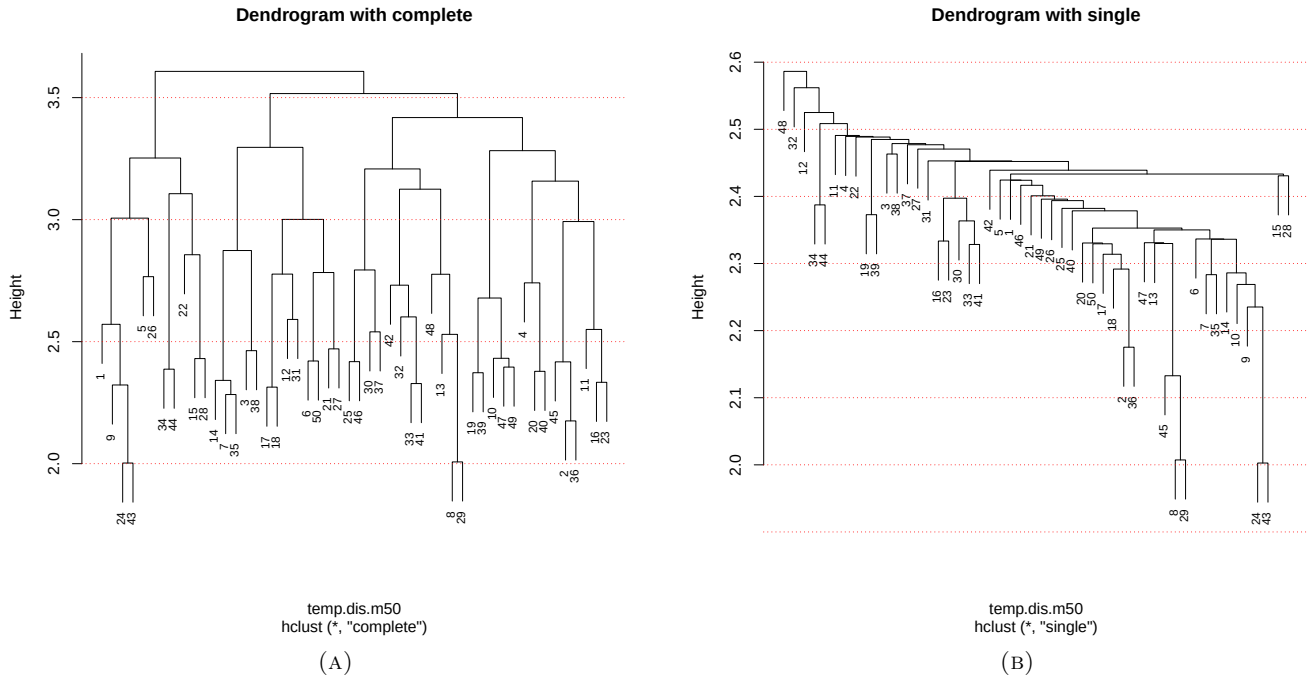
4. An example with larger dendrograms

It is clear that comparing dendrograms by observing their graphs will increase geometrically in complexity as function of the number of elements (entities) from which the dendrograms are obtained. To illustrate that here we created and compared dendrograms obtained for 50 entities using methods "single" and "complete" –which we saw in the previous examples that were the methods which gave larger differences. `Box 4` presents the R calculations.

```
-------------------------------------------------------------------------------------
# Box 4. Constructing and comparing dendrograms for 50 entities.
# Let's obtain two dendrograms for 50 entities
# A matrix for 50 entities having 50*50=2500 values
> set.seed(1959)
> temp.m50 <- matrix(runif(n=2500), nrow=50, ncol=50, dimnames=list(c(1:50), c(1:50)))
> temp.dis.m50 <- dist(temp.m50) # The distance matrix
# Now the cluster using method="single")
> temp.m50.hc.sin <- hclust(temp.dis.m50, method="single")
# Plot that dendrogram
> plot(temp.m50.hc.sin, cex=0.75) # In Figure 5 (B)
# Now the cluster using method="complete")
> temp.m50.hc.com <- hclust(temp.dis.m50, method="complete")
# Plot that dendrogram
> plot(temp.m50.hc.com, cex=0.75) # In Figure 5 (A)
# Now obtain the topologies of both dendrograms:
# For the complete method:
> temp.m50.top.com <- dend.topo(temp.m50.hc.com)
> names(temp.m50.top.com$original)
[1] "V1"     "V2"     "clu"    "class"  "height"
> nrow(temp.m50.top.com$original) # How many groups
[1] 49
> table(temp.m50.top.com$original$class) # Table by sizes
 2  3  4  5  6  8  9 11 12 13 14 26 39 50
19  9  6  3  2  2  1  1  1  1  1  1  1  1

# and for the single method
# For the complete method:
> temp.m50.top.sin <- dend.topo(temp.m50.hc.sin)
> nrow(temp.m50.top.sin$original) # How many groups
[1] 49
> table(temp.m50.top.sin$original$class) # Table by sizes
 2  3  4  5  6  7  8 13 19 20 21 22 23 24 25 26 27 29 30 35 36 37
10  4  3  4  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
38 40 42 43 44 45 47 48 49 50
 1  1  1  1  1  1  1  1  1  1
```

```
# Finally, compare the two dendrogram's topologies
> temp.m50.cd <- comp.dend(temp.m50.top.com, temp.m50.top.sin)
> names(temp.m50.cd)
[1] "basic"        "topo.summary" "identical"    "height.stats"
[5] "objects"      "call"

# First note that dendrograms x and y are respecively
> temp.m50.cd$objects
[1] "temp.m50.top.com" "temp.m50.top.sin"
> temp.m50.cd$call
comp.dend(x = temp.m50.top.com, y = temp.m50.top.sin)

> temp.m50.cd$basic # The basic information
    n.comp    n.ident   likeness w.likeness
48.0000000 11.0000000  0.2291667  0.0177063

# Have a look at the summary of the topology:
> temp.m50.cd$topo.summary
                2         3 4 5 6 7 8 9 11 12 13 14 19 20 21 22
n.in.x  19.0000000 9.0000000 6 3 2 0 2 1  1  1  1  1  0  0  0  0
n.in.y  10.0000000 4.0000000 3 4 1 1 1 0  0  0  1  0  1  1  1  1
n.ident 10.0000000 1.0000000 0 0 0 0 0 0  0  0  0  0  0  0  0  0
p.ident  0.5263158 0.1111111 0 0 0 0 0 0  0  0  0  0  0  0  0  0
        23 24 25 26 27 29 30 35 36 37 38 39 40 42 43 44 45 47 48
n.in.x   0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
n.in.y   1  1  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1  1
n.ident  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
p.ident  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
        49
n.in.x   0
n.in.y   1
n.ident  0
p.ident  0

# Now let's examine the statistics for the heights
> temp.m50.cd$height.stats
           n   Mean.x   Mean.y       S.x       S.y      p.value
All       49 2.690357 2.373975 0.3896692 0.1196362 1.195678e-06
Identical 11 2.282466 2.274522 0.1567103 0.1566887 3.408931e-01

# And the identical clusters:
> temp.m50.cd$identical
   id class id.x id.y height.x height.y     cluster
1   1     2    1    1 2.002694 2.002694    {24, 43}
2   2     2    2    2 2.007293 2.007293     {8, 29}
3   3     2    3    4 2.175313 2.175313     {2, 36}
4   4     2    4    7 2.283380 2.283380     {7, 35}
5   5     3    6    5 2.322573 2.235193 {9, 24, 43}
6   6     2    7   11 2.328380 2.328380    {33, 41}
7   7     2    8   16 2.333486 2.333486    {16, 23}
8   8     2   10   22 2.372767 2.372767    {19, 39}
9   9     2   12   25 2.387364 2.387364    {34, 44}
```

```
10 10     2   17   33 2.430713 2.430713    {15, 28}
11 11     2   19   38 2.463163 2.463163     {3, 38}
> temp.m50.cd$objects # Which is x and which is y
[1] "temp.m50.top.com" "temp.m50.top.sin"
> temp.m50.cd$call
comp.dend(x = temp.m50.top.com, y = temp.m50.top.sin)
-------------------------------------------------------------------------------
```



(A) (B)

FIGURE 5. Dendrograms for 50 entities obtained from the same distance matrix but with two different clustering algorithms, "complete" in panel (A), dendrogram "`x`" in `Box 4` and "single" in panel (B), dendrogram "`y`" in `Box 4`.

In `Box 4` we created and analyzed dendrograms from the same distance matrix for 50 entities labeled by the numbers $1, 2, \cdots, 50$. Properties of the dendrograms are labeled as "`x`" for the one obtained with the method "complete" –panel (A) in Figure 5 and as "`y`" for the one obtained with the method "single" –panel (B) in Figure 5.

By visually examining the dendrograms in Figure 5 only very general aspects of their topologies are immediately evident, say, that the "complete" method (A; `x`) gives more compact clusters than the "single" one (B; `y`), which gives unbalanced and straggly clusters. This is explained by the fact that the "complete" method uses the maximum distance between pair of entities, while the "single" method uses the minimum distance between pair of entities; see Table 4.1 in p. 79 of Everitt et al. (2011). That also explain why the "complete" and "single" methods tend to give the maximum topological differences.

In contrast, by using the function "`comp.dend()`" of our package we directly obtain estimates of likeness between the two dendrograms. In the "`temp.m50.cd$basic`" component we see that the likeness is $11/48 \approx 0.2292$, i.e., the two dendrograms share approximately 23% of their 48 comparable clusters and the weighted likeness, "`w.likeness`", is approximately 0.0177 because only clusters with sizes 2 and 3 are shared between the two dendrograms, i.e., there are no clusters larger than 3 shared between the dendrograms.

By examining heights at which the clusters are formed, i.e., component "`temp.m50.cd$height.stats`" in `Box 4` we see that the "complete" method (A; `x`) has clusters with a significantly larger mean value, 2.69036 than the mean value for the "single" one (B; `y`) which is 2.37397 (row "`All`" in "`temp.m50.cd$height.stats`"). On the other hand, when only the heights in the 11 shared clusters are examined (row "`Identical`" in "`temp.m50.cd$height.stats`"), the differences between the mean heights are not significant (p-value ≈ 0.34), and the cause of that fact can be found by examining the component "`temp.m50.cd$identical`" which shows that the only difference in height is for the shared cluster {9, 24, 43}, because all the other 10 clusters are of size 2 and, obviously, were obtained at identical height by both clustering algorithms.

With this example of relatively large dendrograms we finish the guide for the "**DendroLikeness**" package. I hope that it will be useful for your research and if you have comments please send them by e-mail to `octavio.martinez@cinvestav.mx`. It is worth noticing that using the functions in the package it is relatively easy to implement re-sampling methods to evaluate the robustness of a dendrogram. I plan to include those facilities in a future version of the package.

References

Everitt BS, Landau S, Leese M, and Stahl D (2011) *Cluster Analysis*. Willey.

Felsenstein J (1978) The number of evolutionary trees. *Systematic zoology*, 27, 27–33.

Morlini I and Zani S (2012) Dissimilarity and similarity measures for comparing dendrograms and their applications. *Advances in Data Analysis and Classification*, 6, 85–105.

R Core Team (2013) *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.r-project.org`.

Vidovic SU (2019) Tree congruence: quantifying similarity between dendrogram topologies. *arXiv preprint arXiv:1909.05387*.