The clustComp package

Alvis Brazma and Aurora Torrente

May 27, 2021

1 Introduction

This document presents an overview to the clustComp package, which is a collection of tools developed for the comparison and the visualisation of relationships between two clusterings, either flat versus flat or flat versus hierarchical. Both situations are addressed by representing clusters as nodes in a weighted bipartite graph, where each layer corresponds to one of the clusterings under comparison, and the edge weights are given by the number of elements in the intersection between any two (connected) clusters.

To illustrate the use of the package, we use the publicly available dataset included in the experiment data package, colonCA, on Bioconductor, which contains an exprSet instance for the Alon et al. (1999) colon cancer data, including 40 tumour samples and 22 normal samples, measured at 2000 probeset. We first install and load the experiment data package, which requires the Biobase package:

```
> library(Biobase)
> library(colonCA)
> data(colonCA)
```

For visualisation purposes, we are going to rename some rownames of some of the dataset, which are too long:

```
> rownames(colonCA)[39:42] <- paste("HSAC07", 0:3)
> rownames(colonCA)[50:53] <- paste("UMGAP", 0:3)
> rownames(colonCA)[260:263] <- paste("i", 0:3)
```

Next, we load the clustComp package using:

```
> library(clustComp)
```

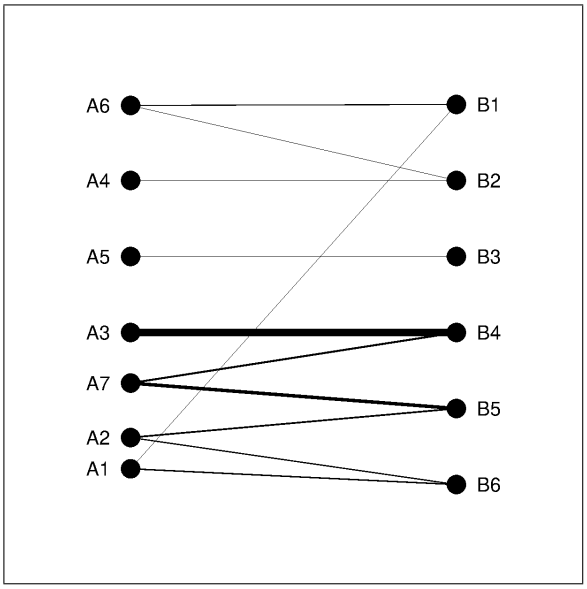
and build the clusterings to be compared. We arbitrarily choose flat clusterings of seven and six clusters, respectively, and two hierarchical trees built with the complete-linkage and the Ward methods, respectively:

```
> set.seed(0)
> # seven flat clusters:
> flat1 <- paste("A", kmeans(exprs(colonCA), 7)$cluster, sep = "")
> # six flat clusters:
> flat2 <- paste("B", kmeans(exprs(colonCA), 6)$cluster, sep = "")
> # dendrograms with 2000 leaves:
> hierar <- hclust(dist(exprs(colonCA)))
> hierar2 <- hclust(dist(exprs(colonCA)), method = 'ward.D')
```

2 Comparison of two flat clusterings

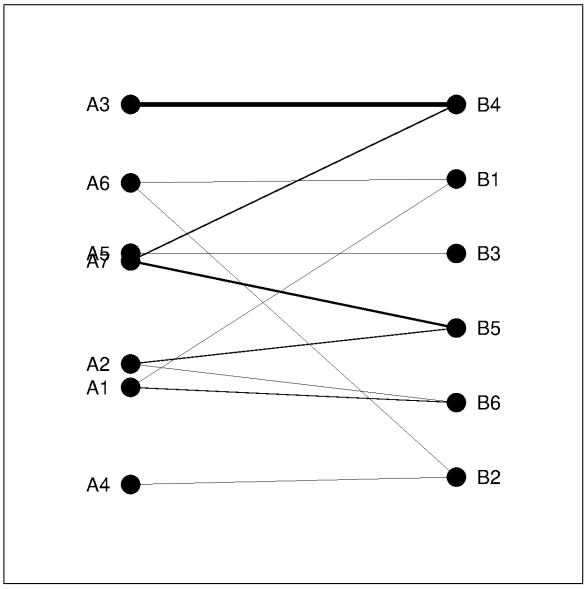
The key algorithm for the comparison of clusterings is the **barycentre algorithm**, which tries to minimise heuristically the number of edge crossings in the bipartite graph, represented by default in a vertical layout. The size of the nodes and the width of the edges can be adjusted by the user, with the arguments `point.sz` and `line.wd`; also a minimum distance `h.min` between consecutive nodes can be set by the user to avoid overlapping.
To compare two flat clusterings, use the function `flatVSflat`, which combines the barycentre algorithm with an additional heuristic consisting in swapping consecutive nodes if that leads to a reduced number of edge crossings. The following run of the function decreases the number of edge crossings from the initial 228422 to the final 9604:

```
> flatVSflat( flat1, flat2, line.wd = 5, h.min = 0.3, greedy = FALSE)

$icross
[1] 695724

$fcross
[1] 25018

$coord1
        A1         A2         A3         A4         A5         A6         A7
-1.7927536 -1.3818898  0.0000000  2.0000000  1.0000000  2.9868421 -0.6654676

$coord2
B1 B2 B3 B4 B5 B6
 3  2  1  0 -1 -2
```



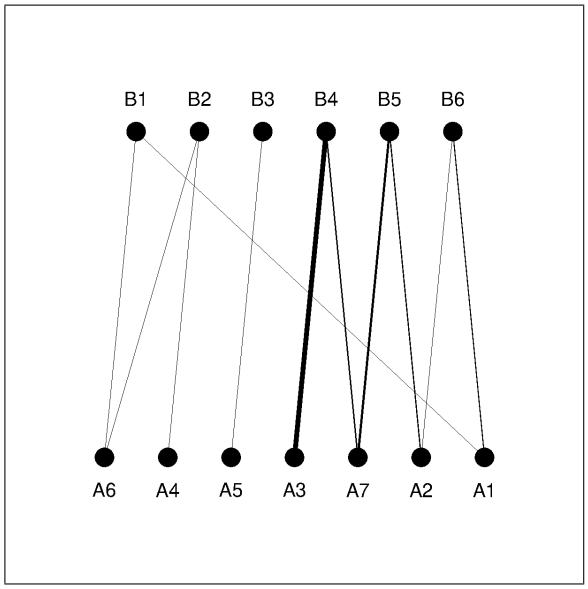
One can easily see that this layout is not the optimal one, but it can be used to find it, by setting an appropriate initial ordering of the nodes:

```
> flatVSflat(flat1, flat2, coord1=c(6,3,4,7,5,1,2),
+     coord2=c(5,1,4,6,3,2), greedy = FALSE)
```



The ordinates of each flat cluster are yielded by the barycentre algorithm; these can be overridden to have evenly distributed nodes (preserving the ordering); also a horizontal layout is allowed:

```
> flatVSflat(flat1, flat2, evenly=TRUE, horiz=TRUE, greedy = FALSE)
```

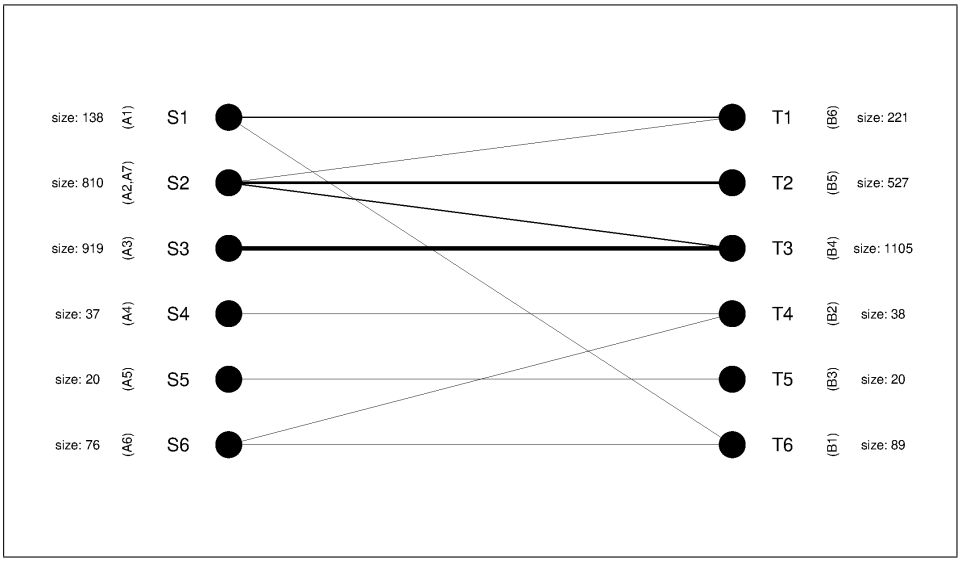


2.1 The greedy algorithm

The **greedy algorithm** constructs a one-to-one mapping between groups of clusters from each partitioning. These new groups are named superclusters; they correspond to the p connected components in the subgraph found by considering only the thickest edge among those which are incident with each node. Ties are not broken at random; instead, all edges with maximum weight are taken into account for constructing the mapping.

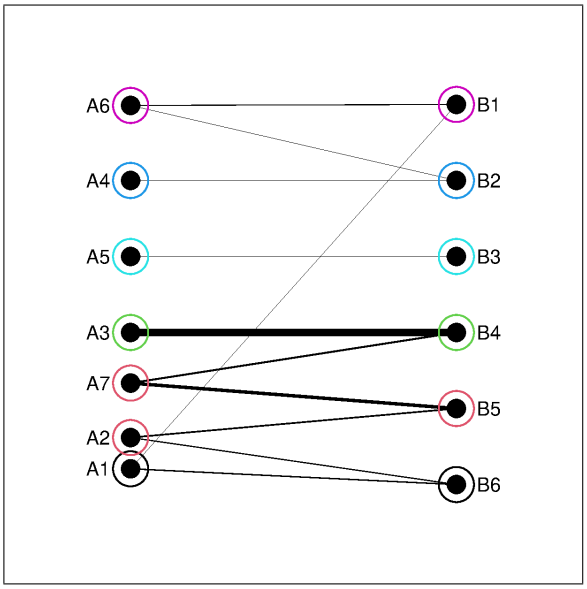
The two sets of p superclusters are assigned new labels: 'S1', 'S2',...,'Sp' for the first clustering, and 'T1', 'T2',...,'Tp' for the second one, as shown below:

```
> myMapping<-SCmapping(flat1, flat2)
```



The initial clusters that form each supercluster are also indicated in the plot, in brackets. After merging them to form the superclusters, all the edges are drawn in the bi-graph (otherwise, it would contain only parallel edges).
Additionally, the mapping between superclusters can be visualised through the function `flatVSflat` by setting the argument `greedy` to TRUE.

```
> flatVSflat( flat1, flat2, line.wd = 5, h.min = 0.3, greedy = TRUE)
```



3 Comparison of flat and hierarchical clusterings

To compare a hierarchical and a flat clusterings we compute cutoffs at different heights of the dendrogram and collapse the resulting branches to form flat clusters, that are compared to the non-hierarchical clustering with the barycentre algorithm, and optionally, with the greedy algorithm.

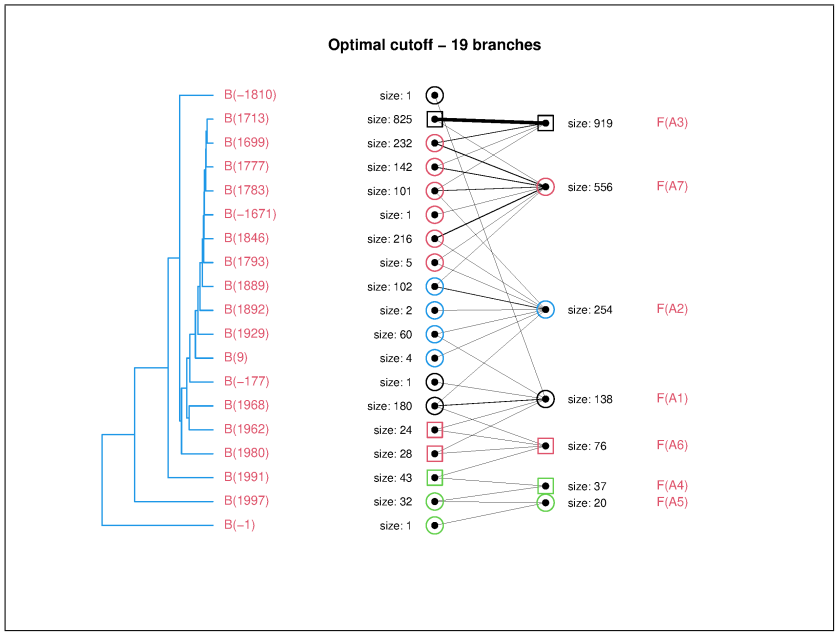
We explore the tree by depth-first search, starting at the root. Each comparison can be sequentially plotted (the default), and then the code needs the interaction of the user to continue, or alternatively, only the final comparison, after all split decisions have been made, is shown. The criterium used to decide whether we should divide a given branch is that splitting that branch yields a better value of a certain scoring function (namely, `"it"`, a more stringent one, based on information theory, or `"crossing"`, based on the layout aesthetics). In other words, we compute a certain score for the 'parent tree' (with the branch under study still collapsed) and for the 'children tree' (with the branch replaced by its descendants), and keep the best one. Note that the number of descendants might be different from 2 if we are using the look-ahead strategy.

The plots, either intermediate or final, display the pruned tree on the left, with branches evenly separated, and the flat clustering on the right, with coordinates given by the barycentre algorithm. Each collapsed branch is labelled with 'B' followed by a number in brackets. That number follows the notation of dendrograms in R: if it is negative, it is a leaf (therefore it cannot be split as it has no children); if it is positive, it indicates the stage at which the given branch was agglomerated when computing the tree (i.e. it has two children). The flat clusters on the right are labelled with 'F' followed by their original label, in brackets. Additionally, a box with as many divisions as branches/clusters indicates the sizes of the corresponding groups.

The following pictures illustrate the use of the function `flatVShier` for the comparison of the hierarchical clustering of the colon data, and the flat clustering with 7 clusters.

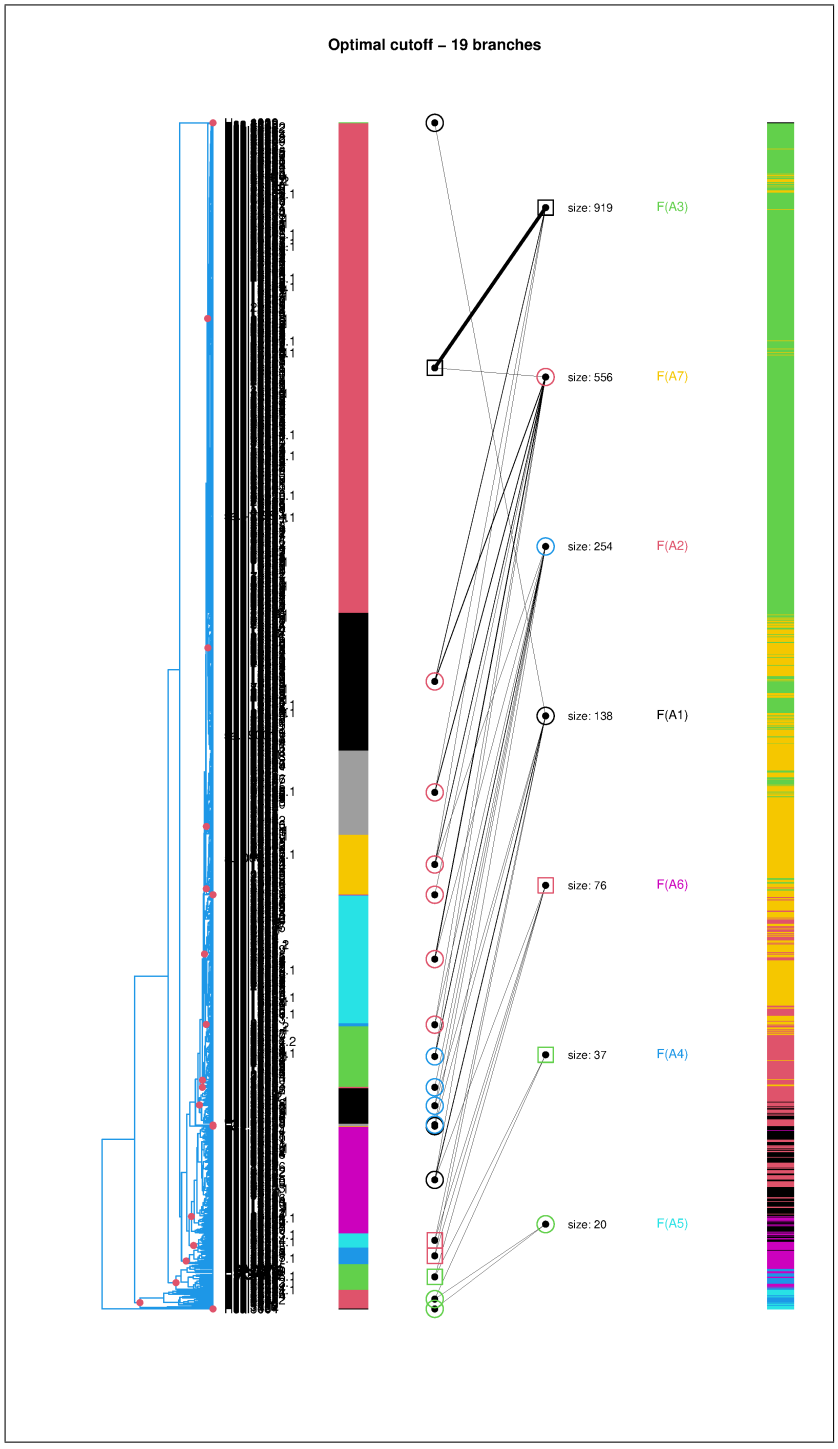
- *Aesthetics-based scoring function, with no look ahead*: the optimal set of cutoffs yields 19 branches; the greedy algorithm computes 7 superclusters, identified in the plot by different colours/symbols.

```
> comparison.flatVShier.1<-flatVShier(hierar, flat1, verbose=FALSE,
+     pausing=FALSE, score.function="crossing", greedy.colours=1:4,
+     look.ahead=0)
```



A more descriptive version of previous figure can be achieved by setting the parameter `expanded` to TRUE; in that case, the full dendrogram is represented:

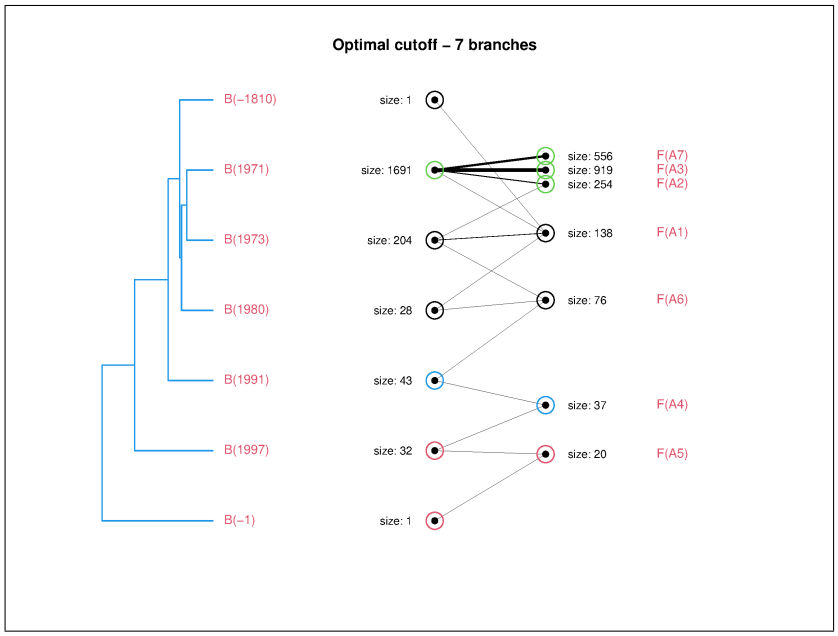
```
> comparison.flatVShier.1<-flatVShier(hierar, flat1, verbose=FALSE,
+     pausing=FALSE, score.function="crossing", greedy.colours=1:4,
+     look.ahead=0, expanded=TRUE)
```



Two coloured bars at both sides of the bi-graph show how genes are distributed across branches and flat clusters.

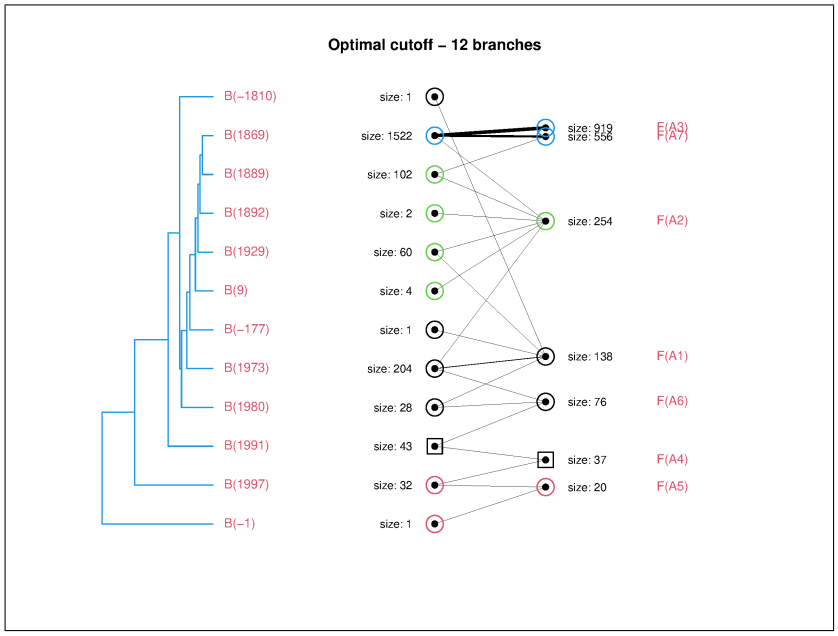
- *Information theoretical-based scoring function, looking one step ahead*: the optimal set of cutoffs yields 7 branches; the greedy algorithm computes 4 superclusters.

```
> comparison.flatVShier.2<-flatVShier(hierar, flat1, verbose=FALSE,
+     pausing=FALSE, h.min=0.2, score.function="it",
+     greedy.colours=1:4, look.ahead=1)
```



- *Information theoretical-based scoring function, looking two steps ahead*: the optimal set of cutoffs yields 12 branches; the greedy algorithm computes 5 superclusters.

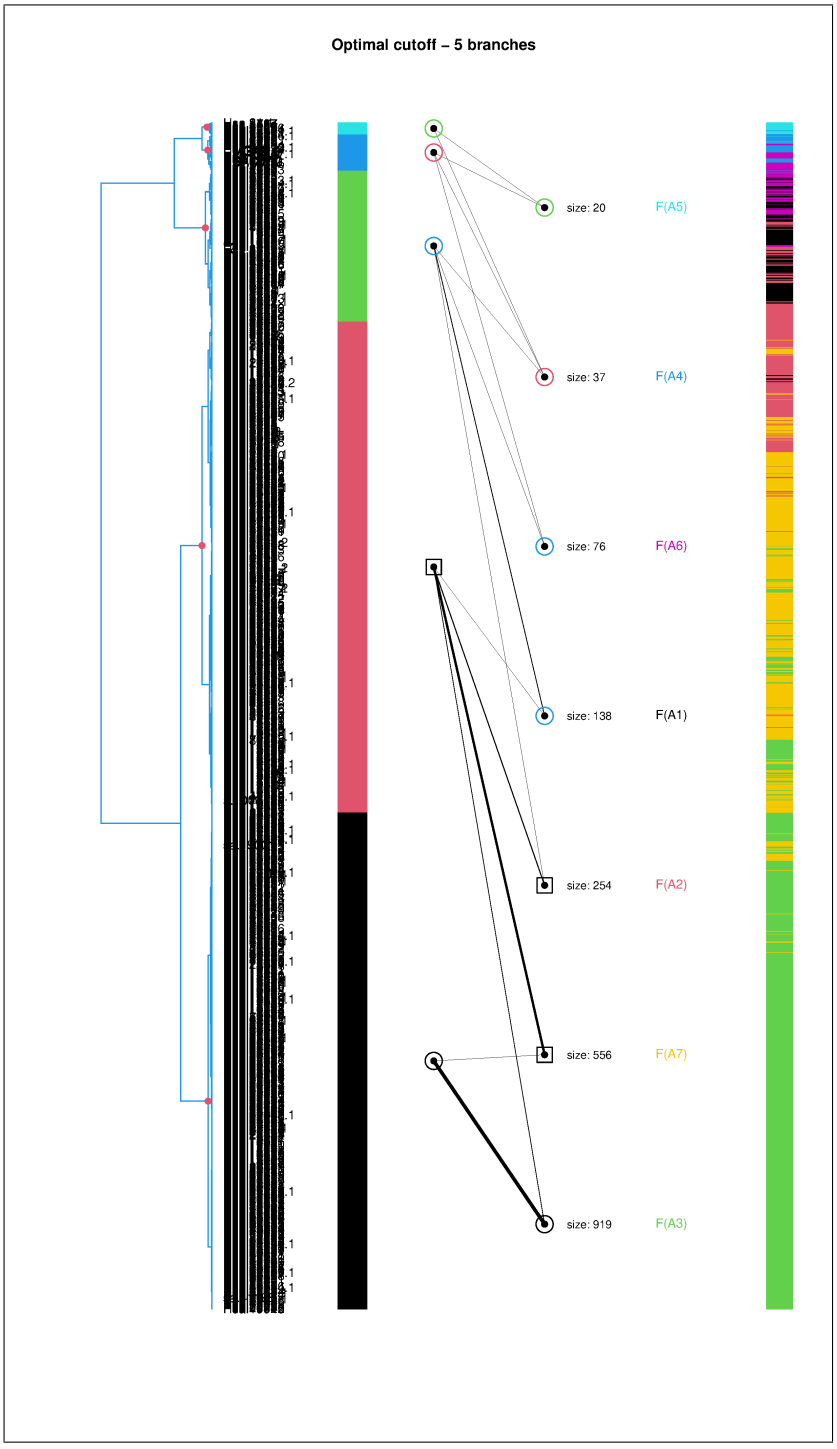
```
> comparison.flatVShier.3<-flatVShier(hierar, flat1, verbose=FALSE,
+     pausing=FALSE, h.min=0.2, score.function="it",
+     greedy.colours=1:4, look.ahead=2)
```



The intermediate steps can be visualised by setting the argument `pausing` equal to TRUE (it requires the interaction of the user to show all the iterations).

Also, an expanded version of the comparison can be achieved by displaying the whole tree, which helps visualising the size of the clusters.

```
> comparison.flatVShier.4<-flatVShier(hierar2, flat1, verbose=FALSE,
+     pausing=FALSE, h.min=0.2, score.function="it",
+     greedy.colours=1:4, look.ahead=2, expanded=TRUE)
```



In the expanded version, it is also possible to draw the heatmap, by providing the expression data as an argument.

```
> comparison.flatVShier.5<-flatVShier(hierar2, flat1, verbose=FALSE,
+     pausing=FALSE, h.min=0.2, score.function="it",
```

```
+      greedy.colours=1:4, look.ahead=2, expanded=TRUE,
+      expression=exprs(colonCA))
```