Dive a Bit Deeper into Hierachical Clustering

Presenter | Sijie Li

2025.08.25

Fast optimal leaf ordering for hierarchical clustering

Ziv Bar-Joseph[1], David K. Gifford[1,2] and Tommi S. Jaakkola[2]

[1]Laboratory for Computer Science, MIT, 545 Technology Square, Cambridge, MA, 02139, USA and [2]Artificial Intelligence Laboratory, MIT, 545 Technology Square, Cambridge, MA, 02139, USA

Received on February 5, 2001; revised and accepted on March 26, 2001

BIOINFORMATICS ***APPLICATIONS NOTE*** *Vol. 24 no. 5 2008, pages 719–720*
doi:10.1093/bioinformatics/btm563

Gene expression

Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R

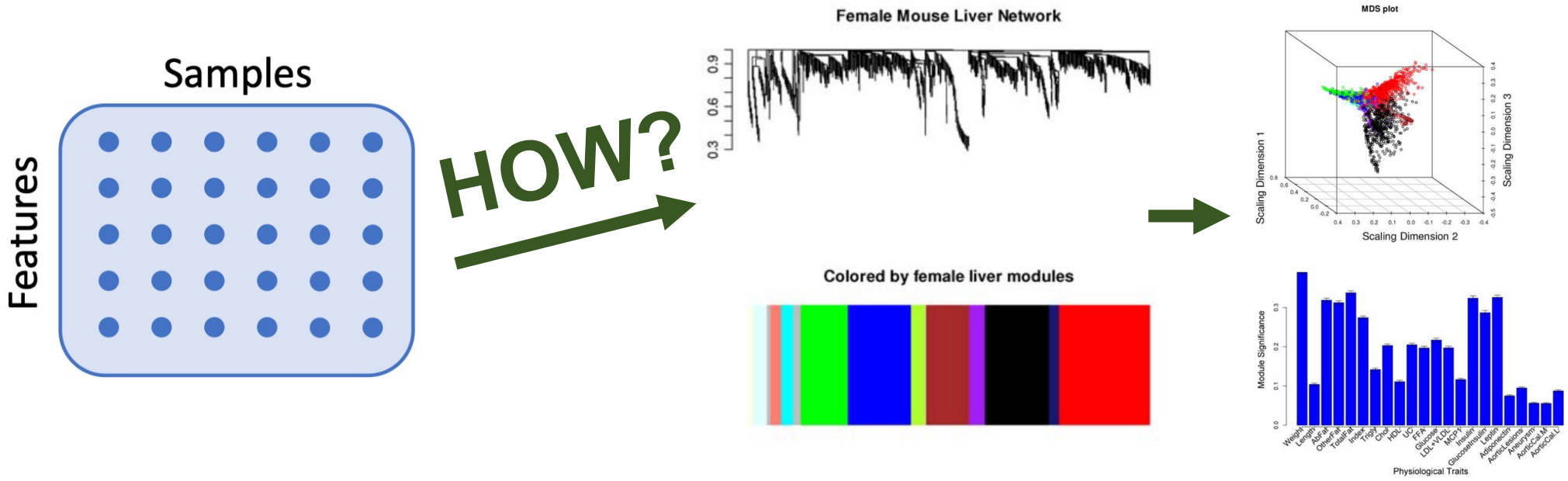
Peter Langfelder[1,†], Bin Zhang[2,†] and Steve Horvath[1,*]

Bioinformatics

Why should we care?

I want to know deeper out of curiosity.

- What's behind the concise codes?
- Widely used in biology → may be useful to know more.
- Utilize to develop useful bioinformatic tools…



* Also, taste papers form unfamiliar realms

(Hovarth et al., 2006)

Why should we care?

Wisdom from the classical maching learning algorithem may sometimes overweighs the fancy deep-learning.

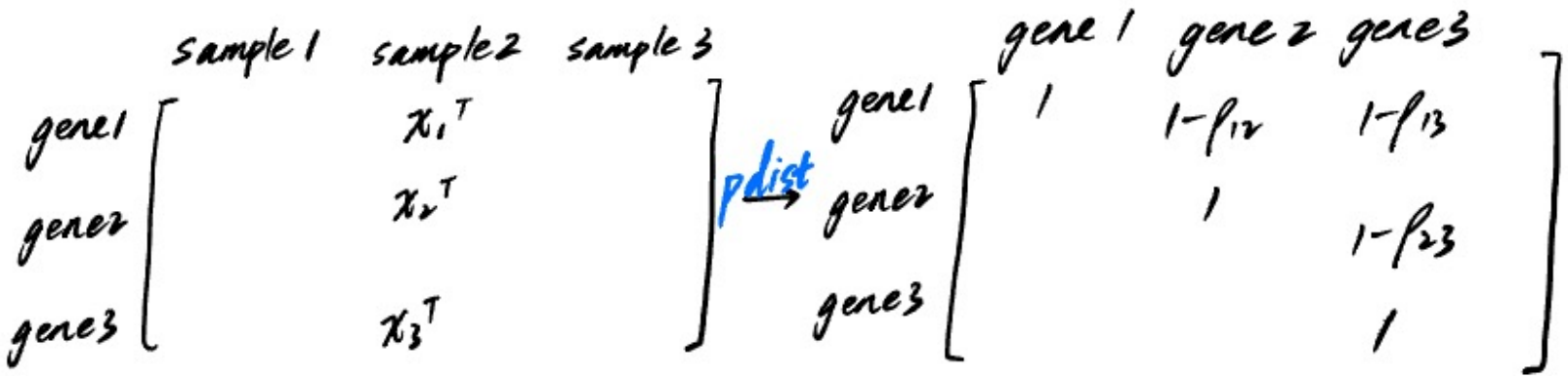
- Absord the wisdom from old papers to get new ideas.
- Make the most of them!

(Anders et al., 2025)

Distance matrix records the dissimilarity.

```
from scipy.spatial.distance import pdist
distance = pdist(rna_bulk,"correlation")
```

$$\rho = \frac{cov(X,Y)}{\sigma_X \times \sigma_Y} = \frac{E[(x_i-\bar{x})(y_i-\bar{y})]}{E[(x_i-\bar{x})^2]^{1/2} \cdot E[(y_i-\bar{y})^2]^{1/2}}$$



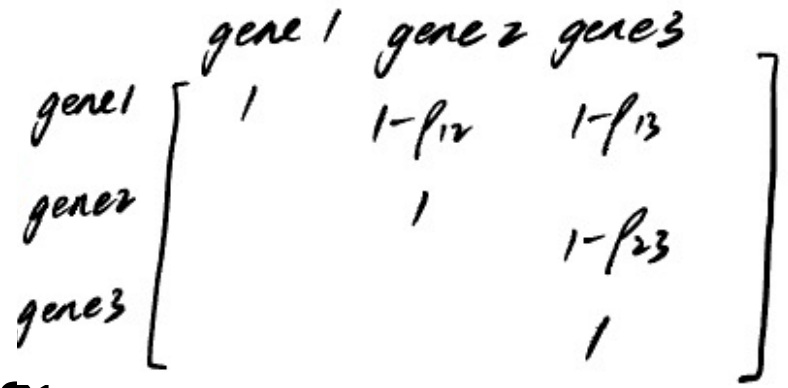
*** Supplement**

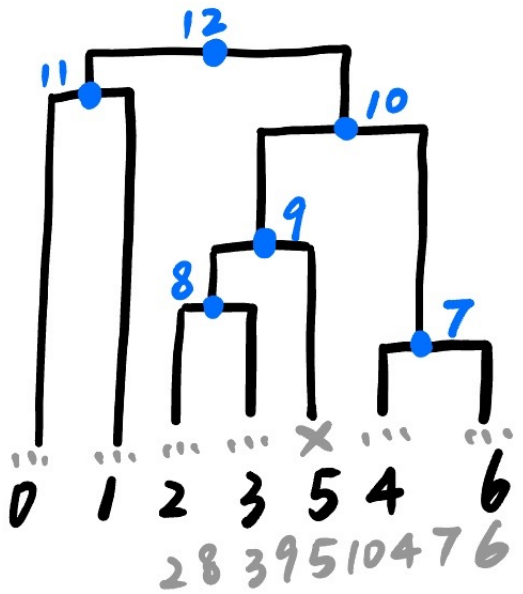
- There are many ways to define “distance”
- Also, other dissimilarity metrics are introduced

e.g. topological overlap dissimilarity measure (dTOM) in WGCNA package. It even considers the scale-free property of the gene network!

Linkage records the hierarchically merging process.

```
link = linkage(distance, "average")
```





node1	node2	merging height	# object
4	6	0.1	2
2	3	0.14	2
8	5	0.2	3
9	7	0.3	5
…			

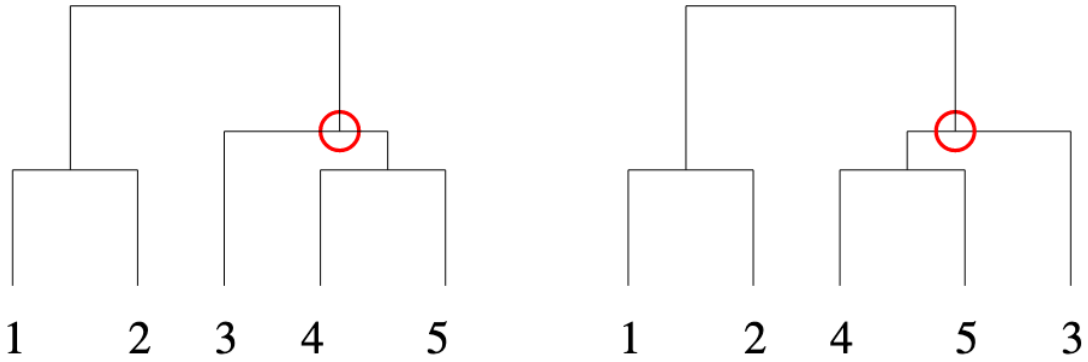
link

Steps

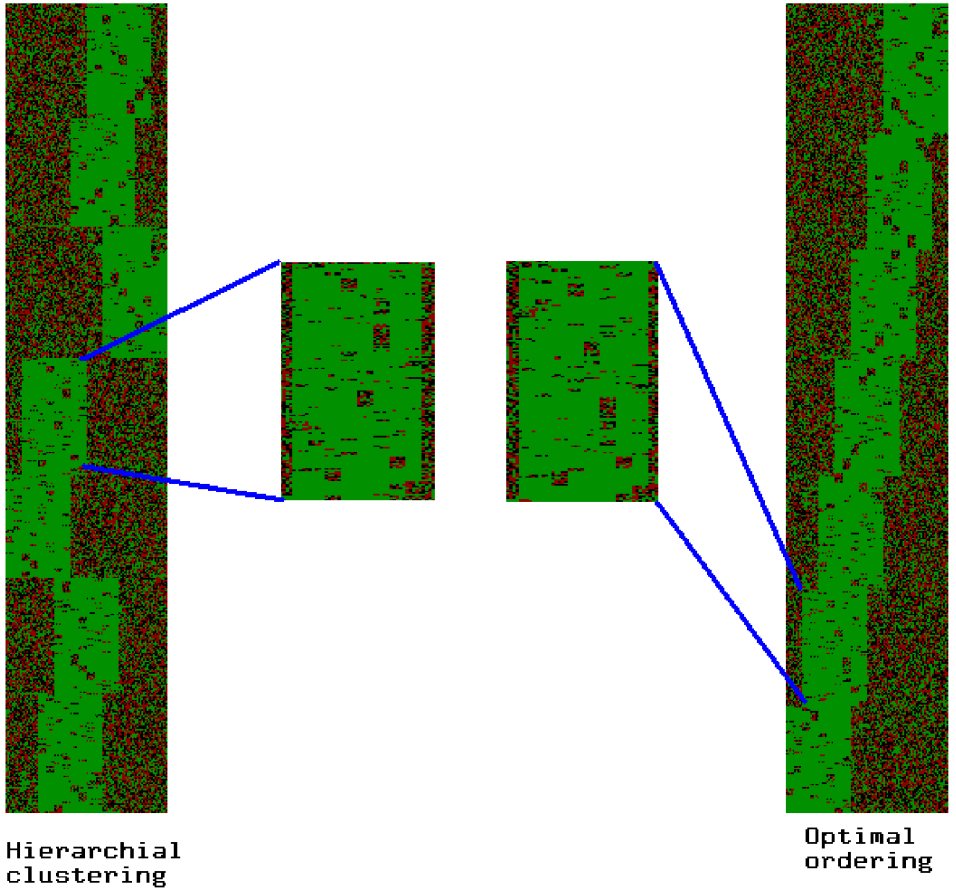
1. Iterate to find minimum
2. merge and replace the distance by the avg
3. [repeat] Iterate to find minimum…

Process of hierarchical clustering—post

There are 2^{n-1} ways to arrange a dendrogram.



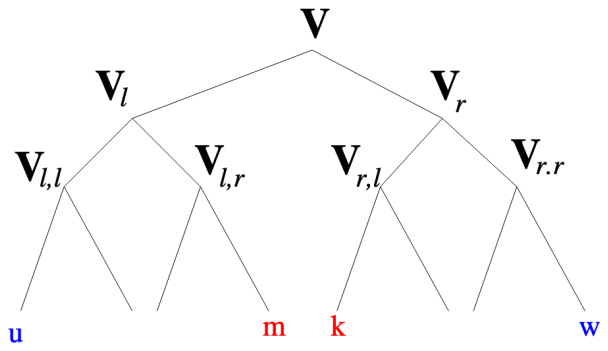
- What is the best ordering?
  - ✓ Max similarity between neighbors
- How to get the best ordering?



(Bar-Joseph et at., 2001)

Dendrogram revealing biological structures is obtained in a bottom up manner.

```
optOrdering(v, S) {
    If |v| = 1 {                    // v has only one leaf
      M(v, u, u) = 0                // u is the only leaf in v
      return M(v, u, u)
    Else                            // v has more than one leaf
      M(v_l, L, R) = optOrdering(v_l, S)      // v_l is the left subtree of v
      M(v_r, L, R) = optOrdering(v_r, S)      // v_r is the right subtree of v
      For all leaves u ∈ v_l {
        For all leaves w ∈ v_r {
          M(v, u, w) = max_{m∈v_{l,r}, k∈v_{r,l}} M(v_l, u, m) + M(v_r, w, k) + S(m, k)
          M(v, w, u) = M(v, u, w)
        }
      }
      return M(v, L, R)             // L, R stands for all possible pairs of leaves from v_l and v_r
}
```



V: subtree/internal node

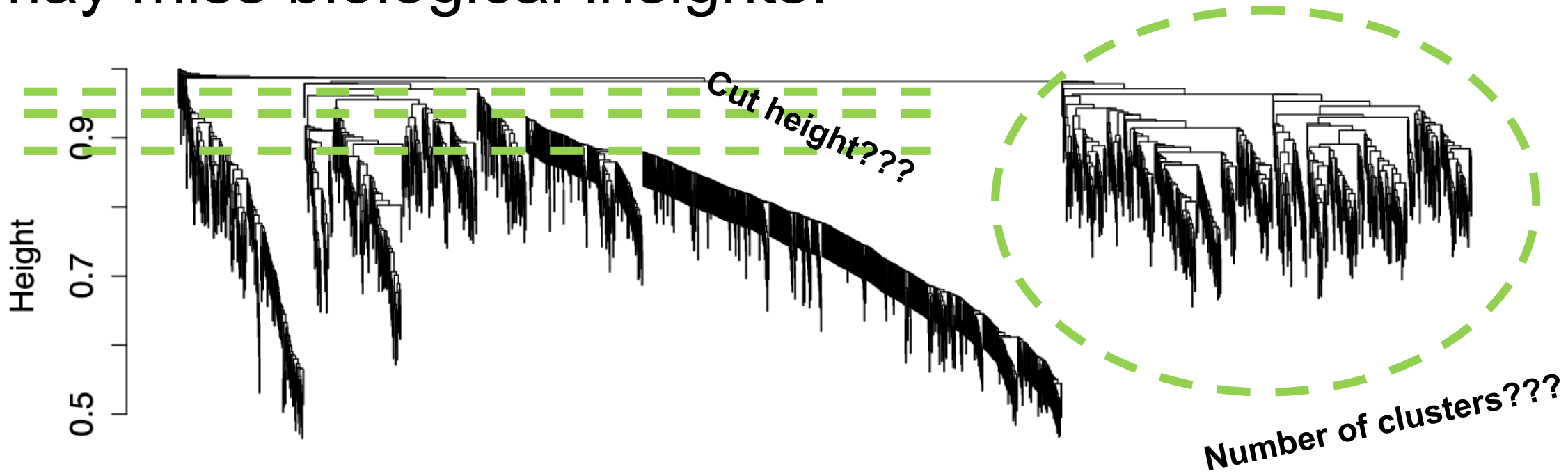
(Bar-Joseph et at., 2001)

Stopping earlier can shorten the time to find the optimal ordering.

$curMax = -\infty$
For all leaves m according to the order of $M(v_l, u, R)$ {
    if $M(v_l, u, m) + M(v_r, w, k_0) + C(v_l, v_r) \leq curMax$ // k_0 is first in the order of $M(v_r, w, L)$
        $M(v, u, w) = curMax$, terminate the search
    For all leaves k according to the order of $M(v_r, w, L)$
        if $M(v_l, u, m) + M(v_r, w, k) + C(v_l, v_r) \leq curMax$
            break // terminate the inner loop
        if $curMax < M(v_l, u, m) + M(v_r, w, k) + S(m, k)$
            $curMax = M(v_l, u, m) + M(v_r, w, k) + S(m, k)$
    } // end of the inner loop
} // end of the outer loop
$M(v, u, w) = curMax$

(Bar-Joseph et at., 2001)

Specifying cut height or cluster number manually may miss biological insights.

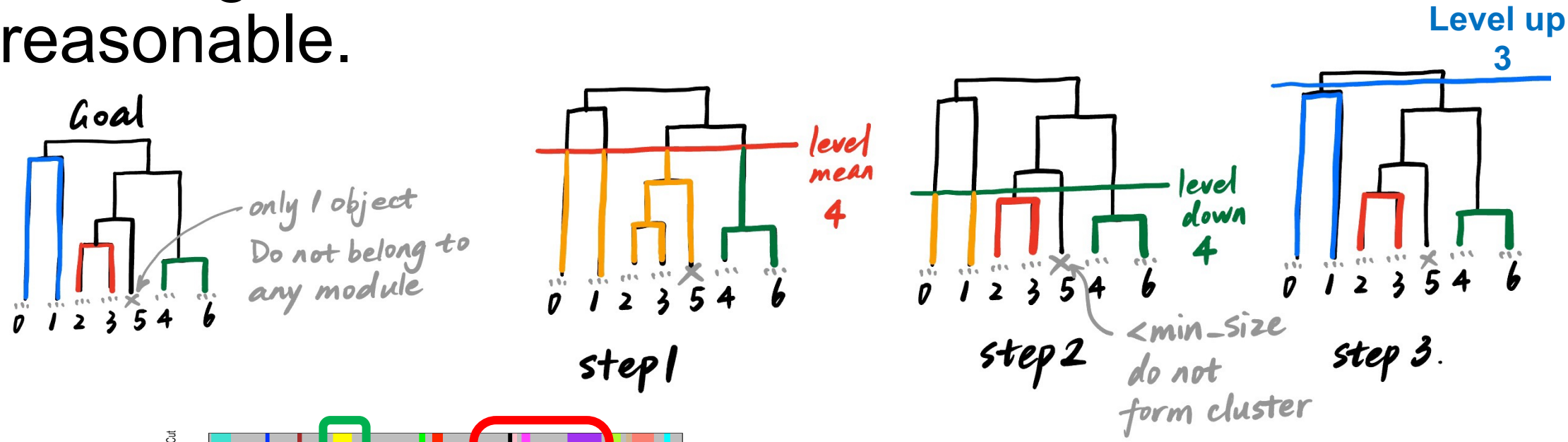


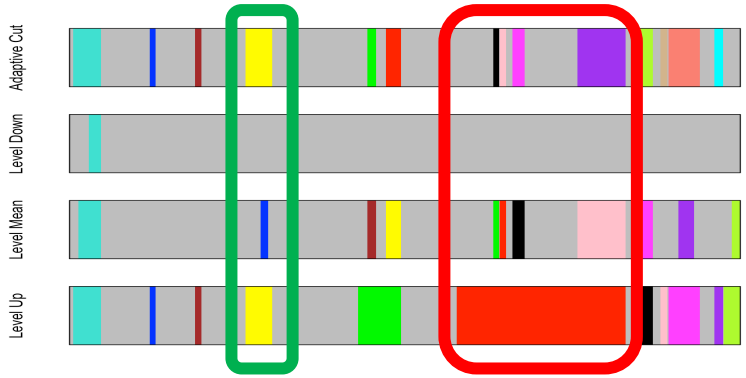
- Manually specifying parameters places the burden of choosing good settings on the user! E.g. adjusting the resolution when doing single cell clustering 😕
- Makes the whole process hard to automate.

(Horvath et at., 2008)

To get clusters!—Dynamic Tree

Cut height can be softened to make clusters more reasonable.

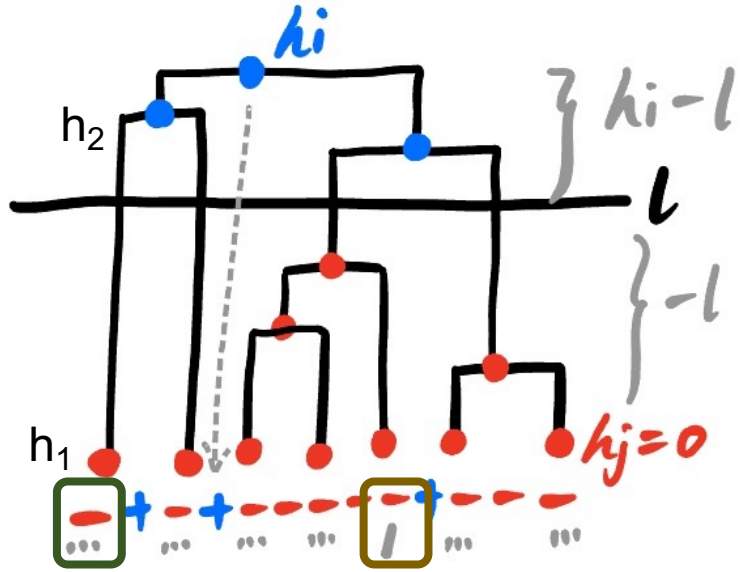




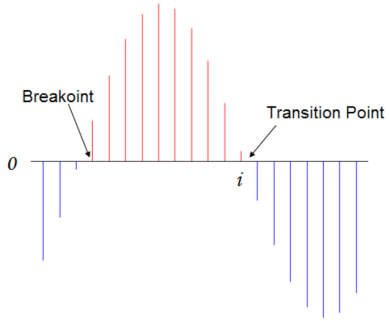
$$
\begin{aligned}
l_m &= \frac{1}{n}\sum_{i=1}^{n} h_i\,, \\
l_u &= \frac{1}{2}(l_m + max\{h_1, h_2, ..., h_n\})\,, \\
l_d &= \frac{1}{2}(l_m + min\{h_1, h_2, ..., h_n\})\,.
\end{aligned}
$$

Reproduce DynamicTreeCut

```
def TreeCut(H,l,min_size):
```



Many subnode No subnode



- Nodes here including merging nodes
- # Nodes in the cluster = # **continuous** **minus-**
- If # > min_size → form a cluster

🤔**Order of H = [h_1,h_2,…]?**

- `hclust()` and `linkage()` only do return the merging process
- Order is the **same as dendrogram**!

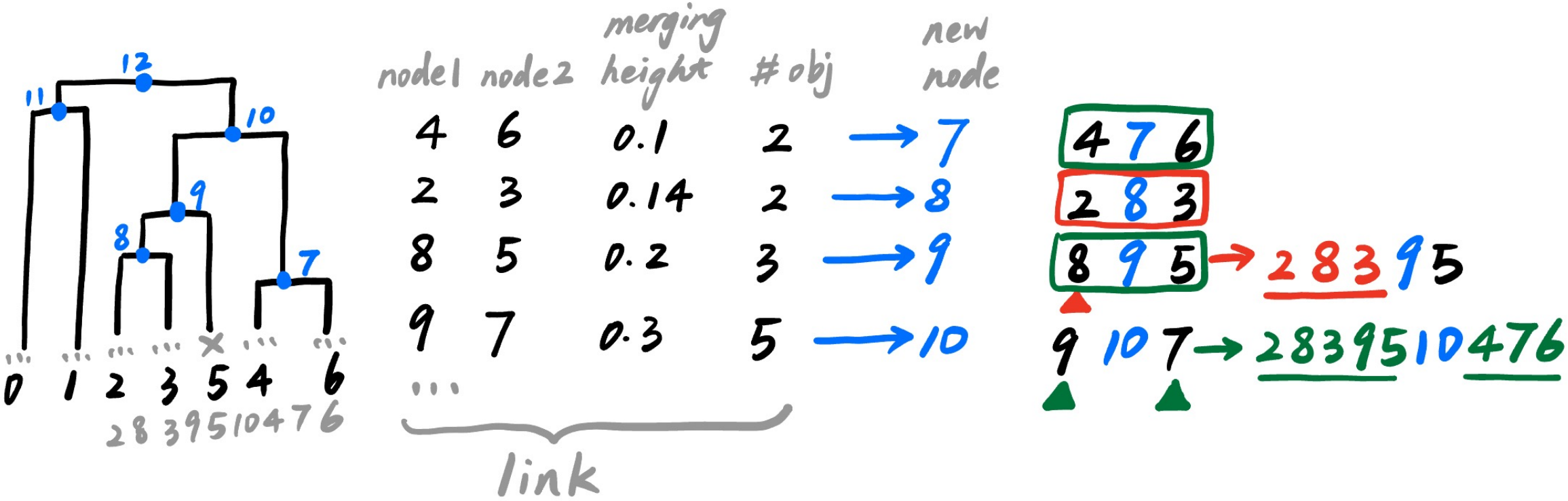
2 You're right to be confused. The only explanation that makes sense is that "dendrogram height sequence" has an unconventional meaning the authors never managed to mention. There's a possible clue in an implementation comment that says it's "the sum of the dissimilarity measure from the root to the node" http://www.bioinformatics.org/cgi-bin/viewvc.cgi/catch/branches/catch-engine/splitter/splitter.c?r1=482&r2=483&view=patch - Gene

Arduous!
Confusions in the paper!

(From stack overflow)

Reproduce DynamicTreeCut

```
def node_order(link):
```

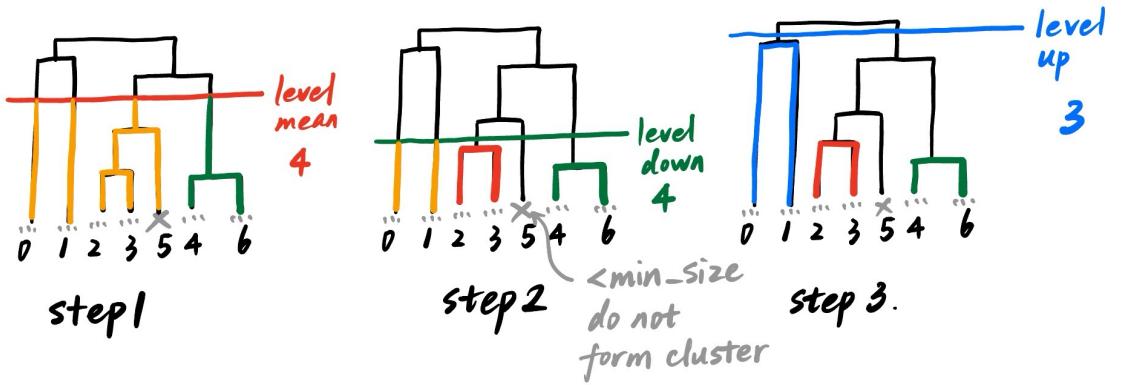


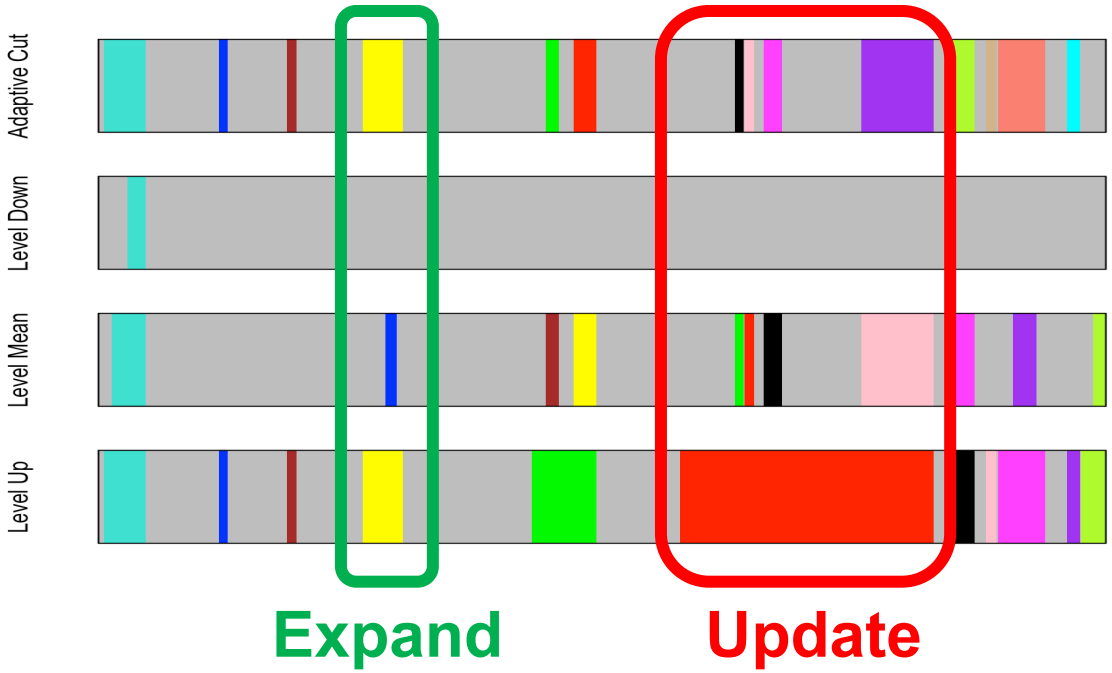
Then **map the height** to the node with link get `height_order`

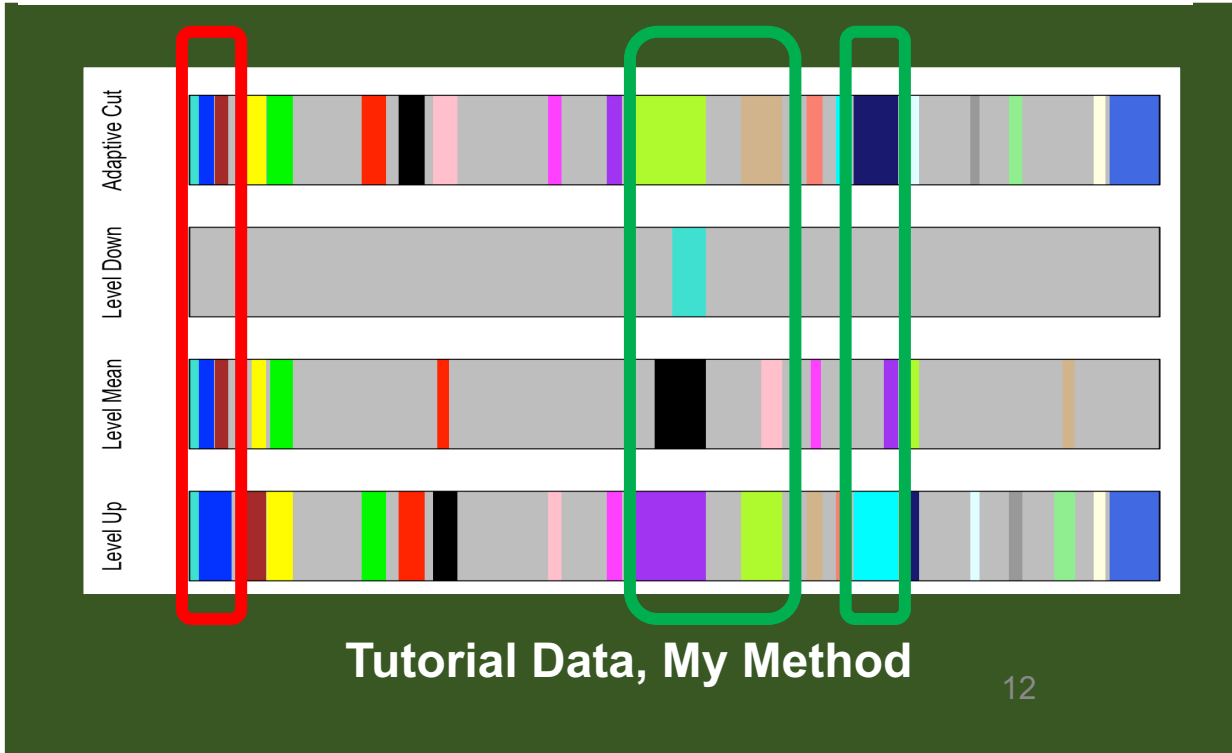
Reproduce DynamicTreeCut

Def AdaptiveTreeCut(H):

- If **≥ 2** sub-clusters in the lower cut level result, **update**
- Or else, **expand** the cluster range



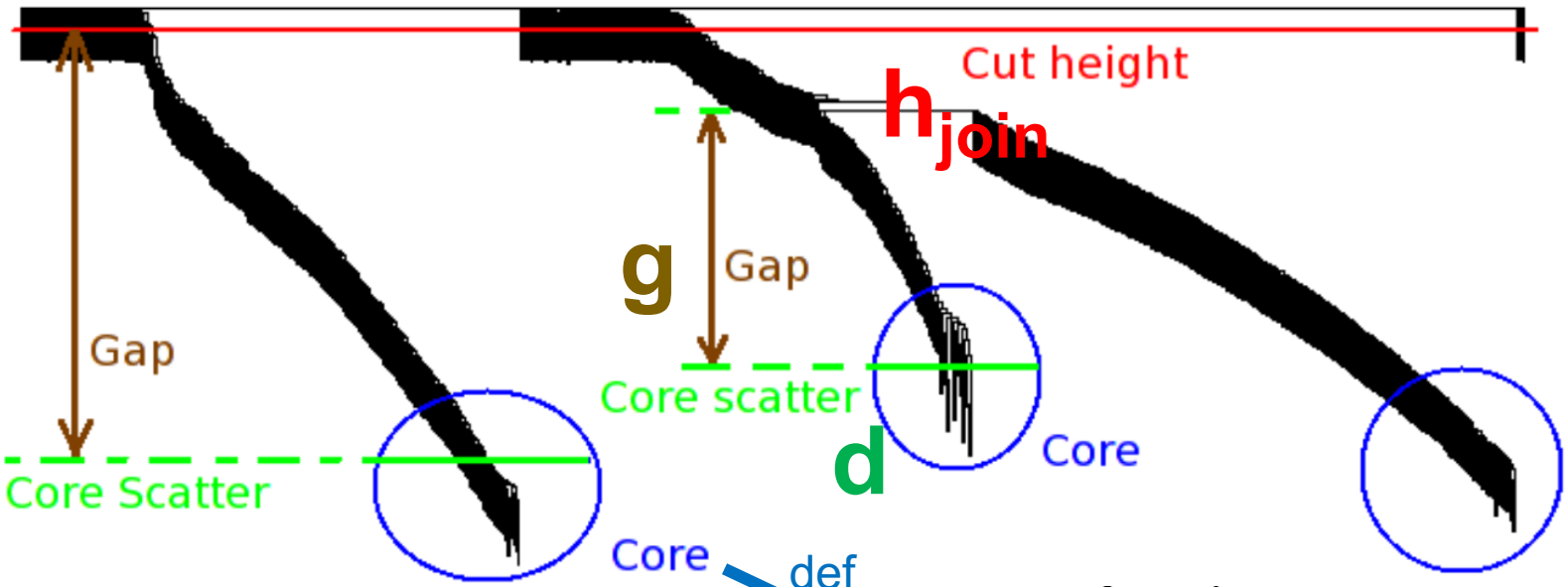




To get clusters!—Dynamic Hybrid

Clusters are defined based on the structure of the dendrogram.

- We can detect clusters easily with our bare eye! Shape matters!



Criteria to be a cluster

- ✓ # leaves ≥ N_0
- ✓ $h_{join} \leq h_{max}$
- ✓ $g \geq g_{min}$
- ✓ $d \geq d_{max}$

Our eyes judge like this!

$$n_c = \min\left\{\text{int}\left(N_0/2 + \sqrt{N - N_0/2}\right), N\right\}$$

(Horvath et at., 2008)

Flat clustering methods can be included for optimization.

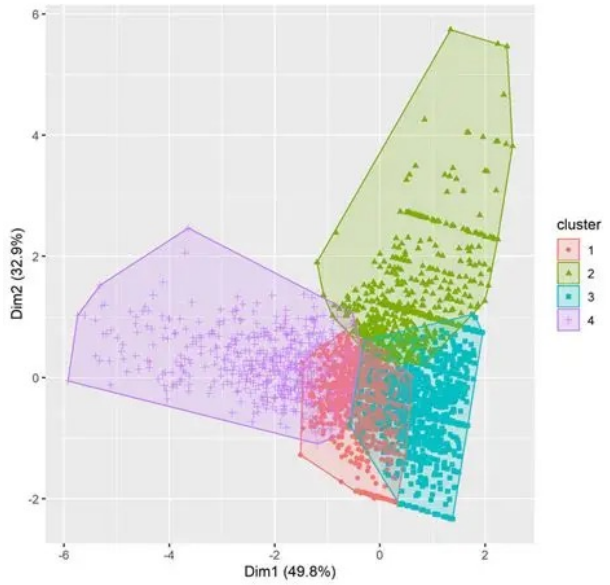
Apply to assign

- unlabeled leaves
- unlabeled tiny cluster (# leaves < N_0)

PAM (partition around medoid)-like method

- Method 1: assign according to avg(leaves-leaves) distance
- Method 2: assign according to leaves-medoid/medoid-medoid distance

PAM

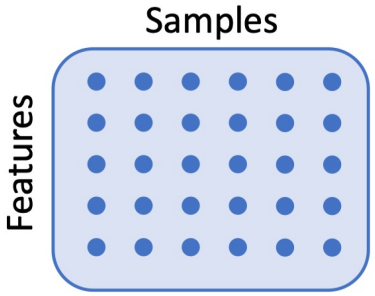


* **Flat clustering methods** always have resolution problem, but **hierarchical clustering** do not have.
Eg. Specify radius for DBSCAN; select number of clusters for k-means

In summary

It’s so satisfying to finally figure all of this out. Hope you feel the same! 😀

Now, what’s happen behind these codes and the principles to get the results are clear!




```
from scipy.cluster import hierarchy
from scipy.spatial.distance import pdist
distance = pdist(rna_bulk,"correlation")
link = hierarchy .linkage(distance, "average")
dn = hierarchy .dendrogram(link)

from dynamicTreeCut import cutreeHybrid
clusters = cutreeHybrid(link, distance)
```