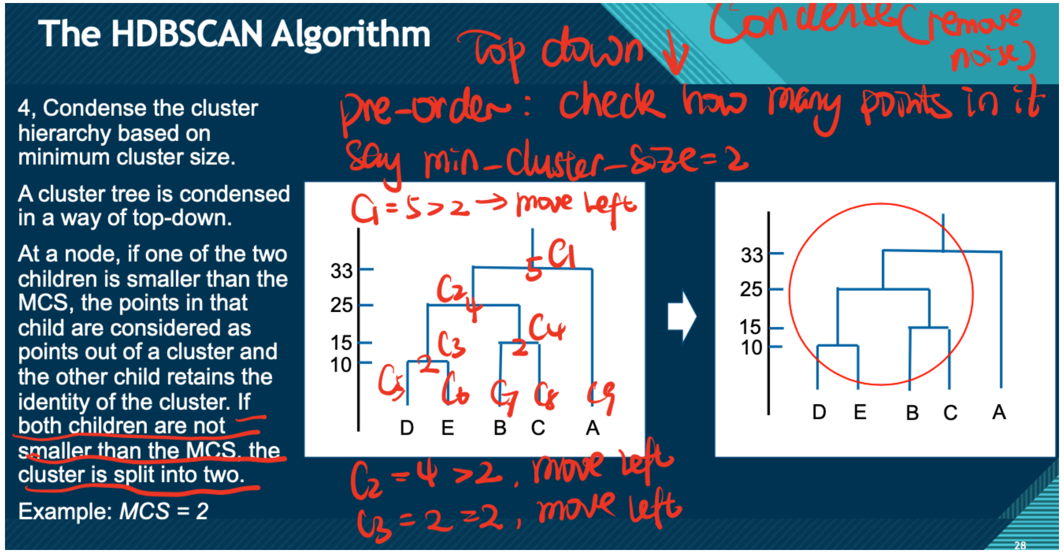
Condense:



The HDBSCAN Algorithm

Top down ↓

Condense (remove noise)

4, Condense the cluster hierarchy based on minimum cluster size.

A cluster tree is condensed in a way of top-down.

At a node, if one of the two children is smaller than the MCS, the points in that child are considered as points out of a cluster and the other child retains the identity of the cluster. If both children are not smaller than the MCS, the cluster is split into two.

Example: *MCS* = 2

pre-order: check how many points in it

say min_cluster_size=2

$C_1 = 5 > 2$ → move left

$C_2 = 4 > 2$, move left

$C_3 = 2 = 2$, move left

* 2n-1nodes -> 标记 （child >= MCS）

* 顺便 记录下 lambda_birth

0. Union Find 的时候，要存下来每个 internal node 下辖几个 Leaf nodes, 这些 leaf nodes 都是谁， 记为 children[]

1. Condense 的时候，层序遍历整棵 hieracia tree，标记出所有 Length(children) < MCS 的 internal node

2. 注意，被标记的 internal node 不一定真的是 noise，最后是不是还得结合产生这个 potential noise 点的 stability 判断

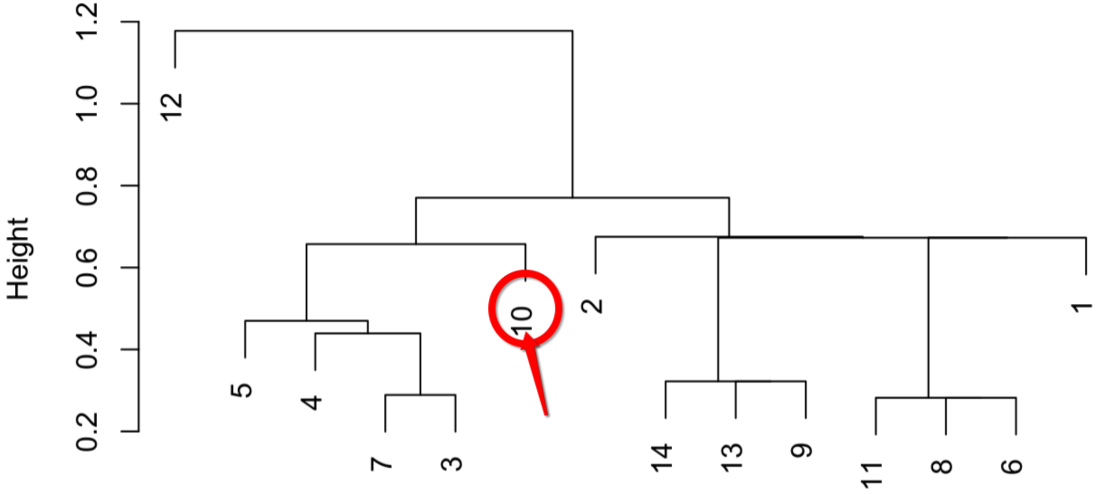
3. Bonus, top down 的时候，可以顺带存下来所有 internal node 的 lambda_birth （不妨令 root 的 birth = root.child 的 birth）

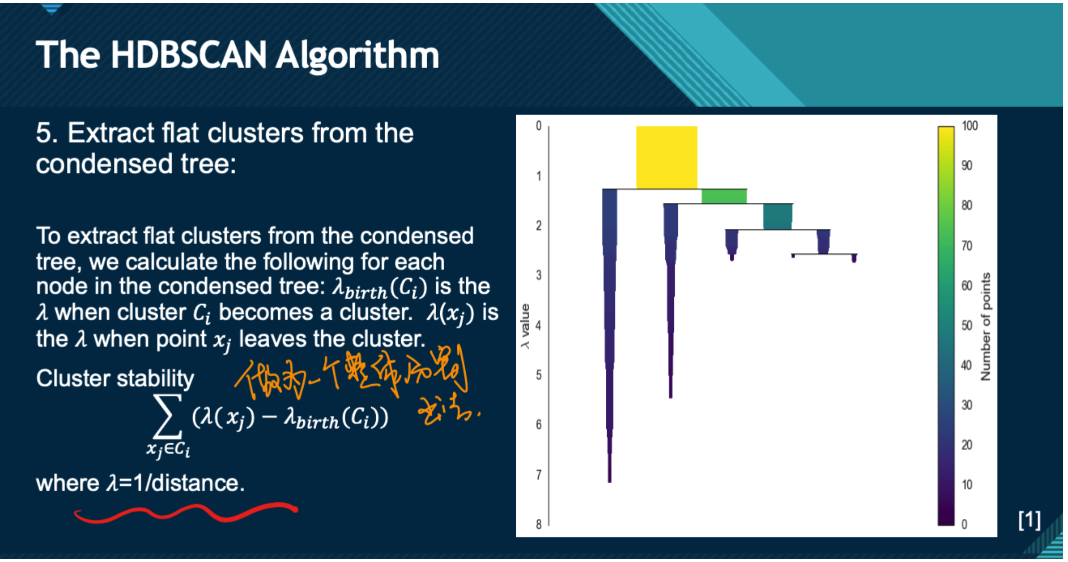
比如 MCS = 3 时， 10 的 parent node，children = [4,3,5,7,10]

Num of left children > 3 不被标记

Num of right child = 1 被标记，但是你现在还不能确定 10 是 noise

Cluster Dendrogram





The HDBSCAN Algorithm

5. Extract flat clusters from the condensed tree:

To extract flat clusters from the condensed tree, we calculate the following for each node in the condensed tree: $\lambda_{birth}(C_i)$ is the λ when cluster C_i becomes a cluster. $\lambda(x_j)$ is the λ when point x_j leaves the cluster.

Cluster stability 作为一个整体而割出去.

$$\sum_{x_j \in C_i} (\lambda(x_j) - \lambda_{birth}(C_i))$$

where λ=1/distance.

[1]

Extract [全部需要 post-order bottom up 整棵树， 不是 condensed tree

1) Calculate Node **lambda_death** for each internal node [左面的求和]

$$\sum_{x_i} \lambda(x_i)$$

Lambda death:

1) Leaf node lambda_death = lambda_birth
2) Internal node，如果两侧 children 都在 condense tree 里
   [不是 potential outlier]：
   Lambda_death = (num_children)* (left_child_lambda_brith)

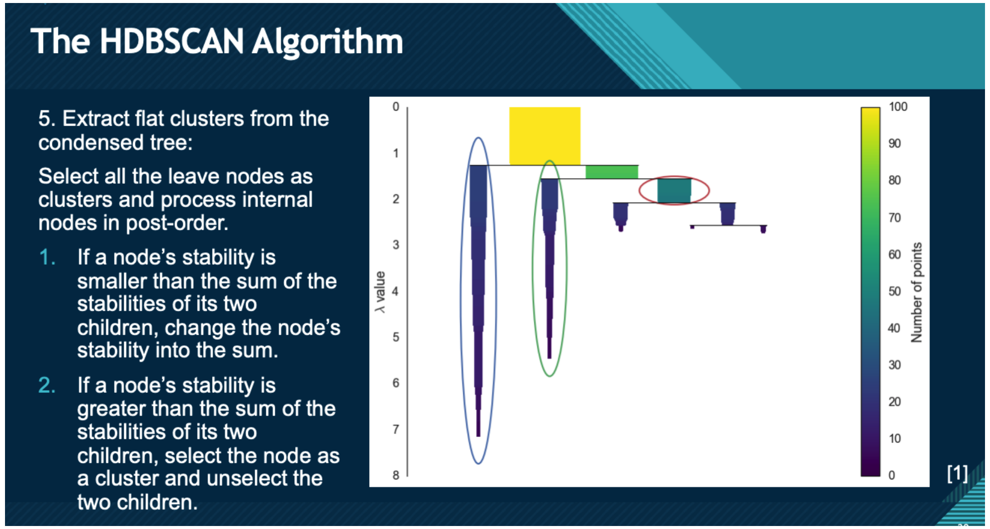
   否则
   Lambda_death = left_child_death + right_child_death

2) Calculate Node **Stability**

$$\sum_{x_i} \lambda(x_i) - \sum_{x_i} \lambda_{birth}(C)$$

上一步的结果 – num_children * lambda_birth

n-1 stability for each internal nodes
n leaf node stability = 0



The HDBSCAN Algorithm

5. Extract flat clusters from the condensed tree:

Select all the leave nodes as clusters and process internal nodes in post-order.

1. If a node's stability is smaller than the sum of the stabilities of its two children, change the node's stability into the sum.
2. If a node's stability is greater than the sum of the stabilities of its two children, select the node as a cluster and unselect the two children.

[1]

3) Exact and find noise

Leaf_node_stability = 0

Internal_node_update_stability = max(stability, left stability +right. stability)

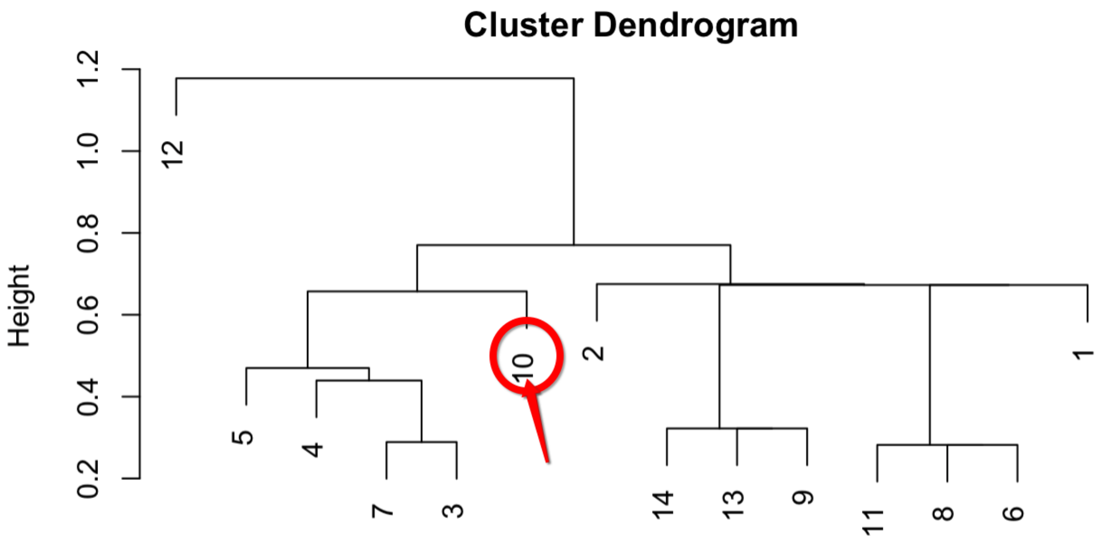
如果 merge 更稳定：

所有的 children 都标记为同一个 cluster

如果 分成两个 cluster 更稳定：

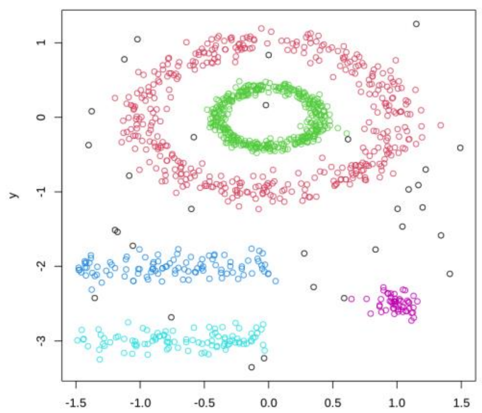
如果 left/right 中的一个被 condense 标记为 potential outlier

这个时候，就可以确定他们是 global noise

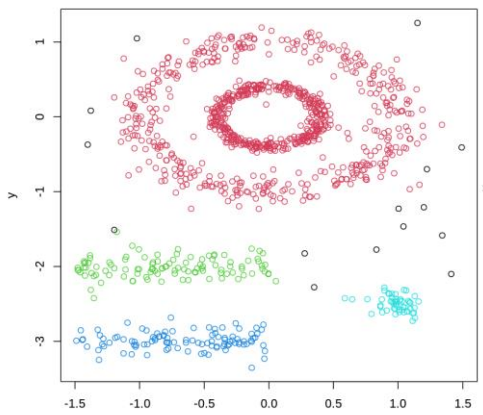


MCS = 3

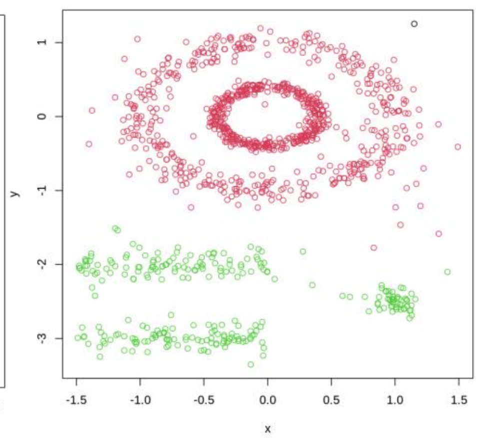
DBSCAN standard lib with eps: 0.15, minPts: 5



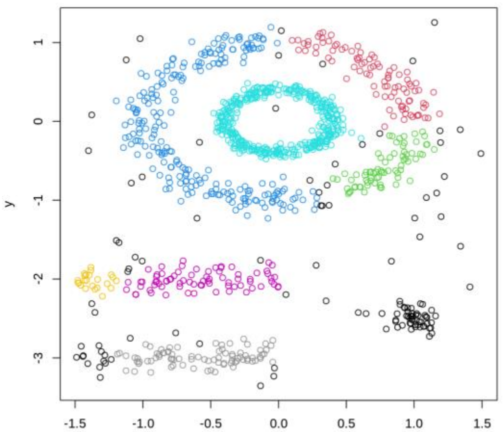
DBSCAN standard lib with eps: 0.25, minPts: 5



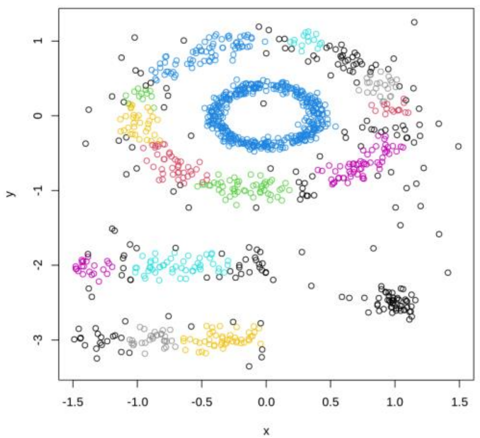
DBSCAN standard lib with eps: 0.45, minPts: 5



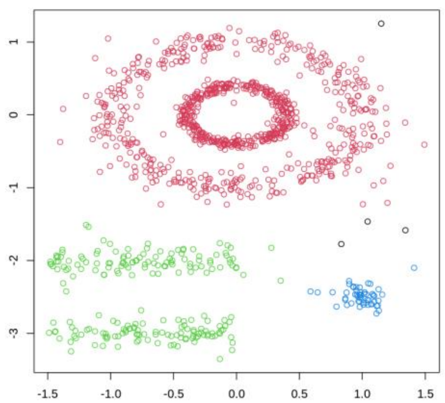
DBSCAN standard lib with eps: 0.15, minPts: 10

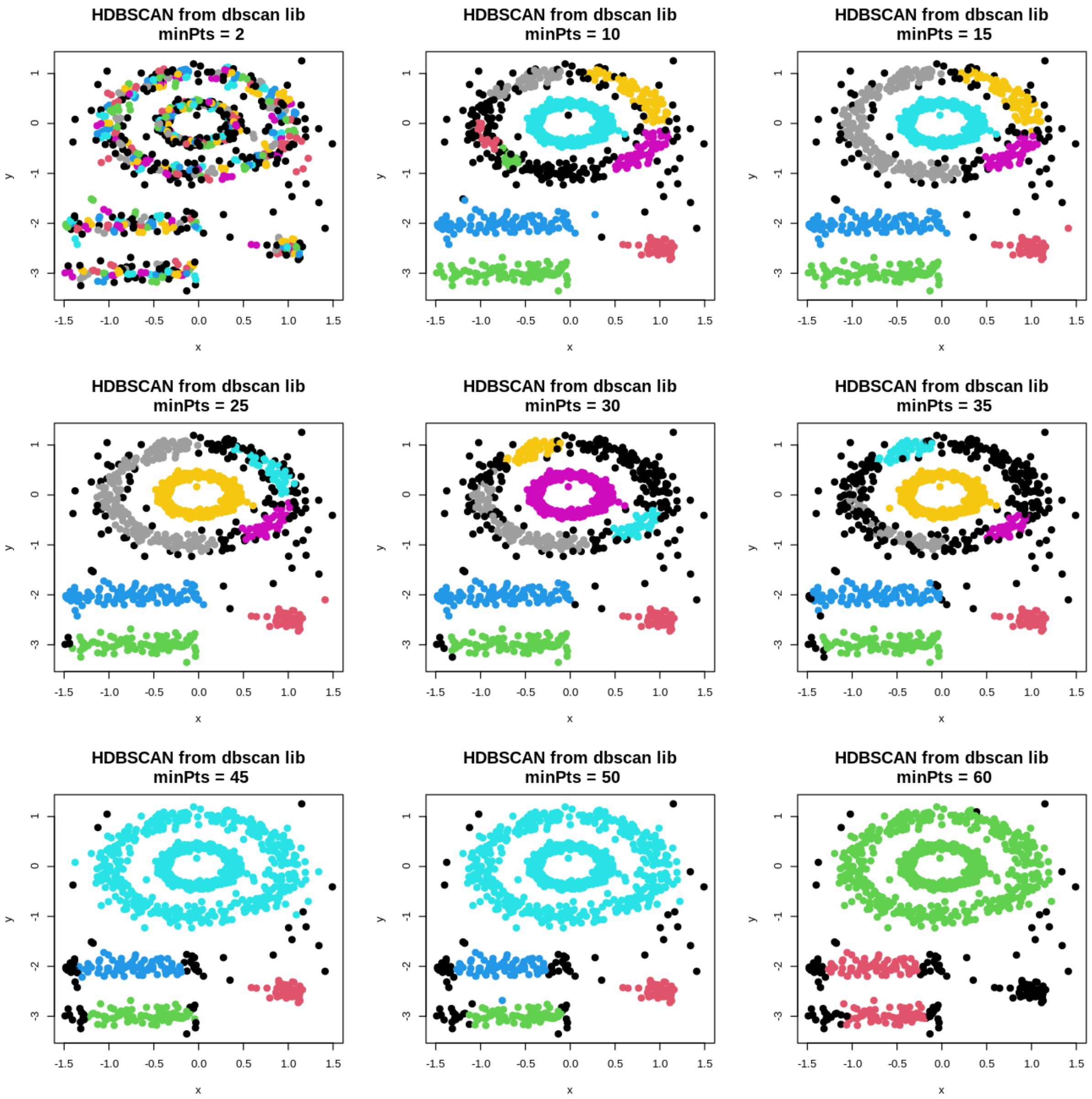


DBSCAN standard lib with eps: 0.15, minPts: 15



DBSCAN standard lib with eps: 0.45, minPts: 15

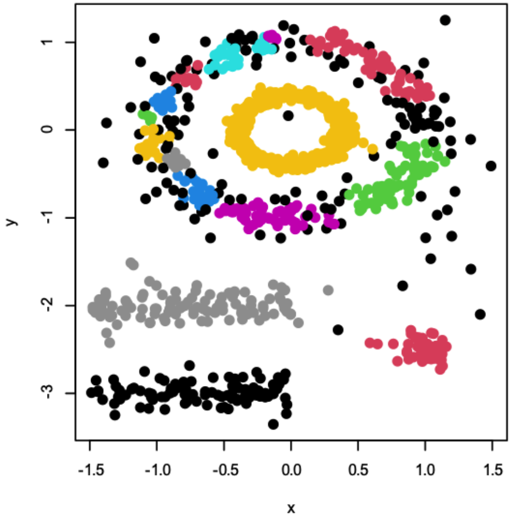




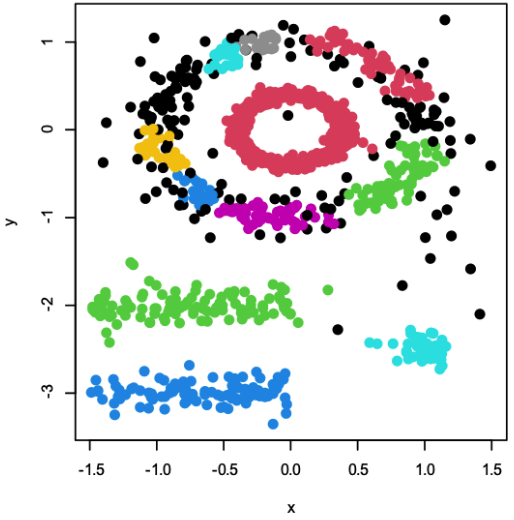
Using the `dbscan` lib’s `hdbscan()` method:

try different MCS (minPts), from 2 to 60;
Non of them shows a reasonable clustering result

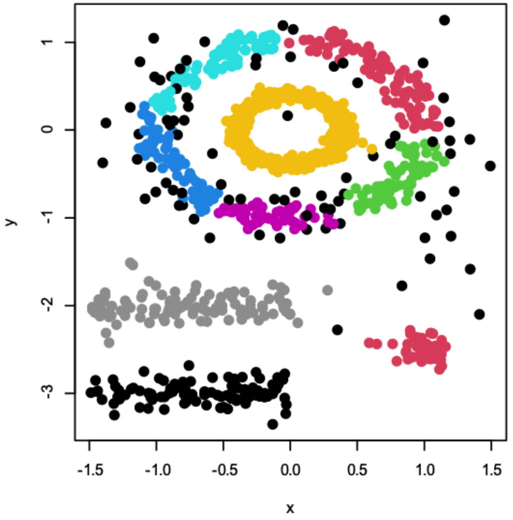
My HDBSCAN Implement
MCS = 5 minPts = 5



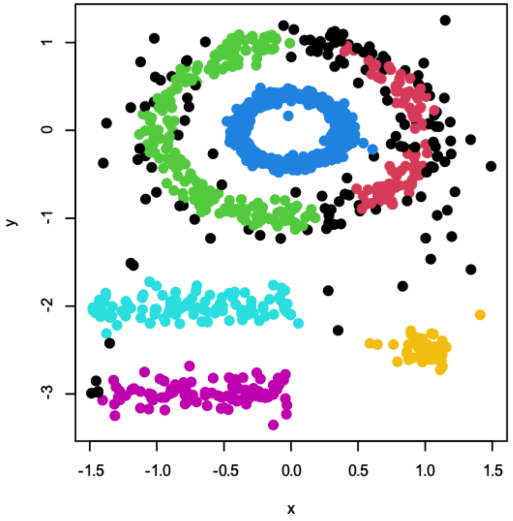
My HDBSCAN Implement
MCS = 15 minPts = 5



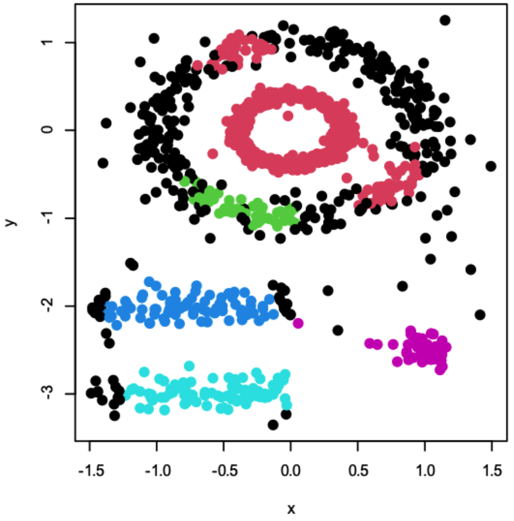
My HDBSCAN Implement
MCS = 50 minPts = 5



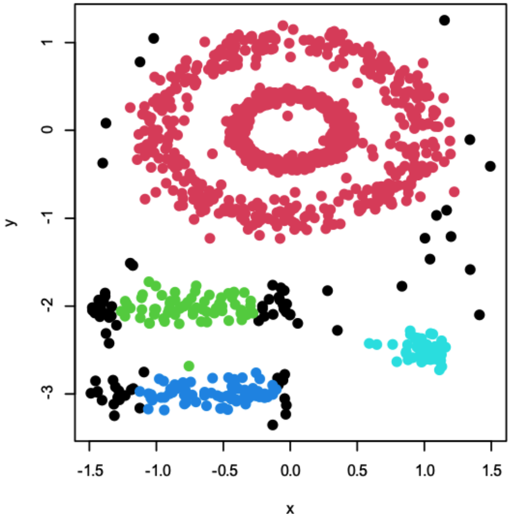
My HDBSCAN Implement
MCS = 50 minPts = 25



My HDBSCAN Implement
MCS = 50 minPts = 40

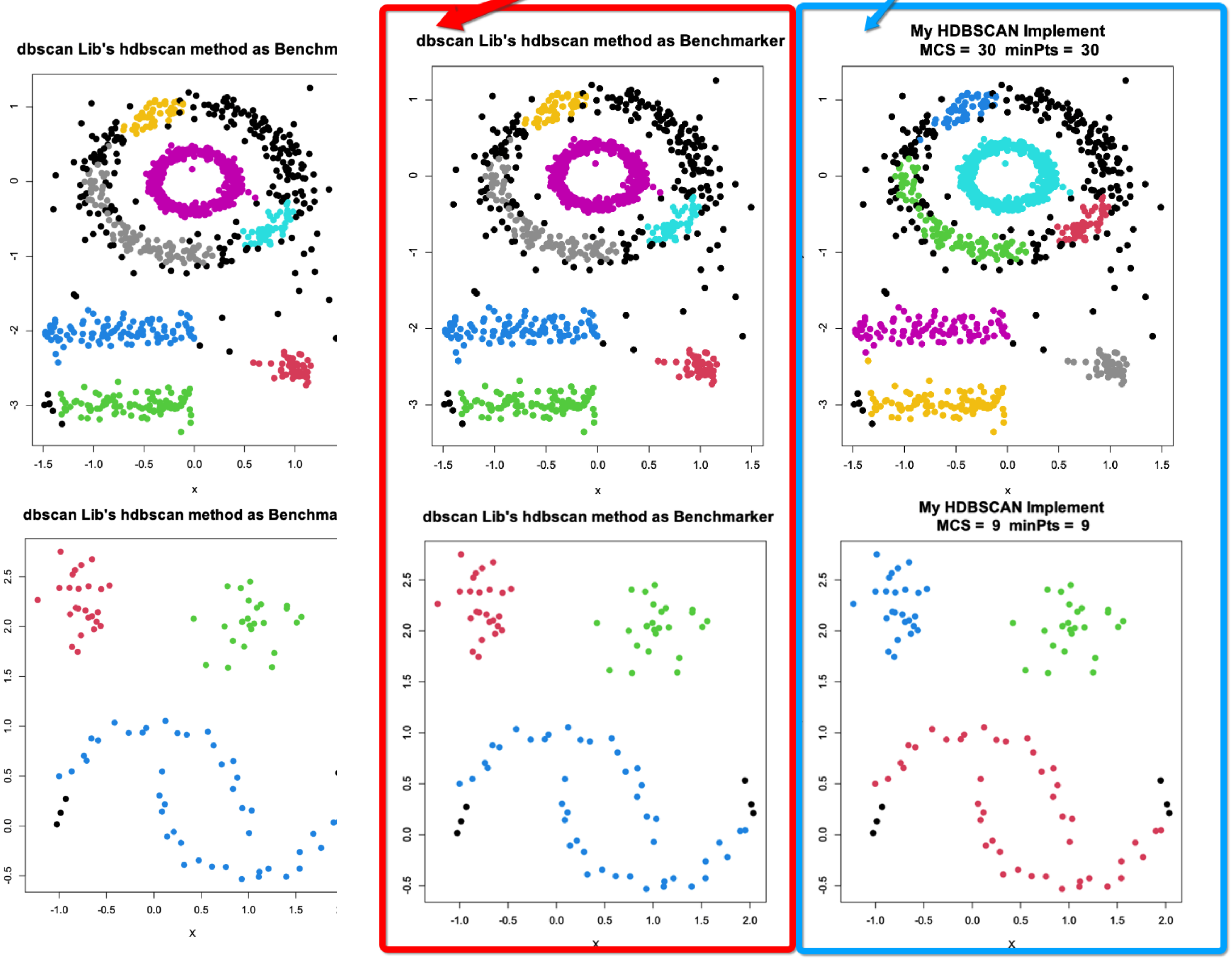


My HDBSCAN Implement
MCS = 50 minPts = 50



Package in CRAN

My implementation


dbscan Lib's hdbscan method as Benchm
dbscan Lib's hdbscan method as Benchmarker
My HDBSCAN Implement
MCS = 30 minPts = 30
dbscan Lib's hdbscan method as Benchma
dbscan Lib's hdbscan method as Benchmarker
My HDBSCAN Implement
MCS = 9 minPts = 9
x