Output of the code clustvarsel

Stepwise (forward/backward) greedy search

-----------------------------------------

Variable proposed Type of step BIC difference Decision

1 X99 Add 147.10967 Accepted

2 X39 Add 131.58678 Accepted

3 X60 Add 323.07781 Accepted

4 X99 Remove 51.05747 Rejected

5 X200 Add 34.32259 Accepted

6 X39 Remove -11.97349 Accepted

7 X39 Add -11.97349 Rejected

8 X99 Remove 103.59784 Rejected

Selected subset: X99, X60, X200

> clust.fit$subset

X99 X60 X200

100 61 201

> clust.fit1

Headlong (forward/backward) search

----------------------------------

Variable proposed Type of step BIC difference Decision

1 X99 Add 147.109669 Accepted

2 X64 Add 50.917093 Accepted

3 X39 Add 12.032581 Accepted

4 X64 Remove -68.637103 Accepted

5 X60 Add 323.077814 Accepted

6 X60 Remove 323.077814 Rejected

7 X204 Add 2.814599 Accepted

8 X204 Remove 2.814599 Rejected

9 X200 Add 6.423350 Accepted

10 X204 Remove -25.084643 Accepted

11 X234 Add 13.622079 Accepted

12 X234 Remove 13.622079 Rejected

13 X191 Add -21.829725 Rejected

14 X234 Remove 13.622079 Rejected

Selected subset: X99, X39, X60, X200, X234

> clust.fit1$subset

X99 X39 X60 X200 X234

100 40 61 201 235

Output of code dbscan

> res <- dbscan::dbscan(nci, eps = 33.8, minPts = 4)

> res

DBSCAN clustering for 64 objects.

Parameters: eps = 33.8, minPts = 4

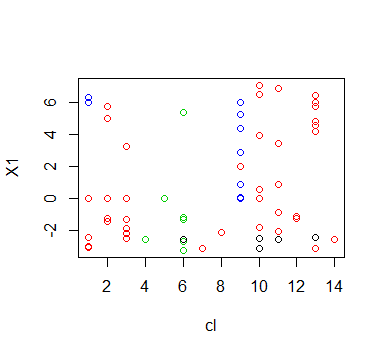
The clustering contains 3 cluster(s) and 5 noise points.

0 1 2 3

5 43 7 9

Available fields: cluster, eps, minPts

The plot of dbscan is not very accurate but I am trying to work on it.



The DBSCAN method proposed consists of computing the the **k-nearest neighbor distances** in a matrix of points. The algorithm calculates the average of the distances of every point to its k nearest neighbors. The value of k is specified by the user and corresponds to **MinPts**. The k-distances are plotted in an ascending order. The aim is to determine the “knee”, which corresponds to the optimal **eps** parameter.

A knee corresponds to a threshold where a sharp change occurs along the k-distance curve. The function **kNNdistplot()** [in **DBSCAN** package] can be used to draw the k-distance plot.