Machine learning

Chapter 10

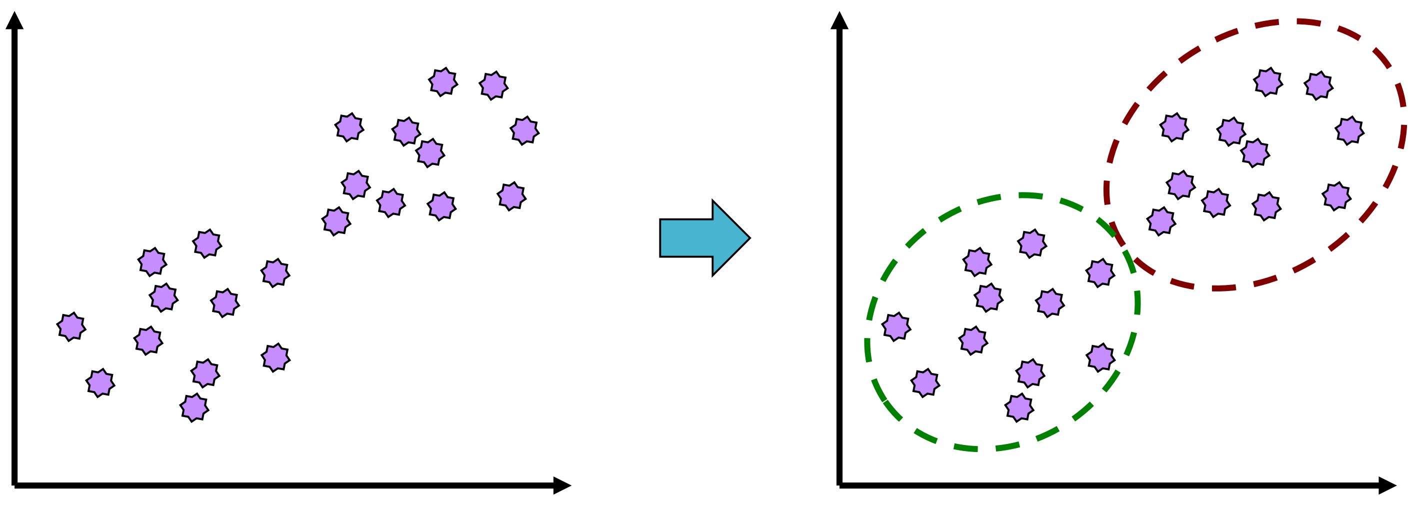
Clustering

Sejong Oh
Bio Information Technology Lab.

Contents

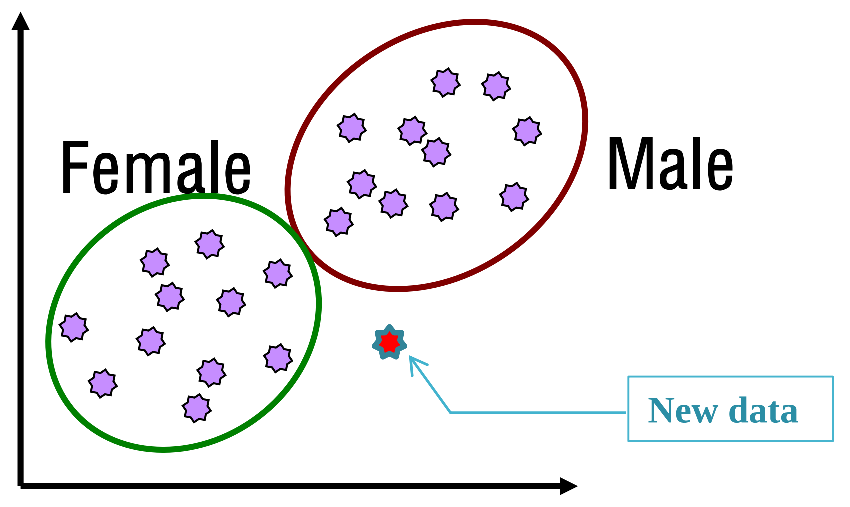
Summary

- Clustering
  - Grouping target data into some category
  - Data in same group has similar characteristics
  - Group points into clusters based on how "near" they are to one another
  - Unsupervised learning



Summary

- Classification
  - Classify new data into one of known category.
  - The category has "label"
  - Supervised learning



Summary

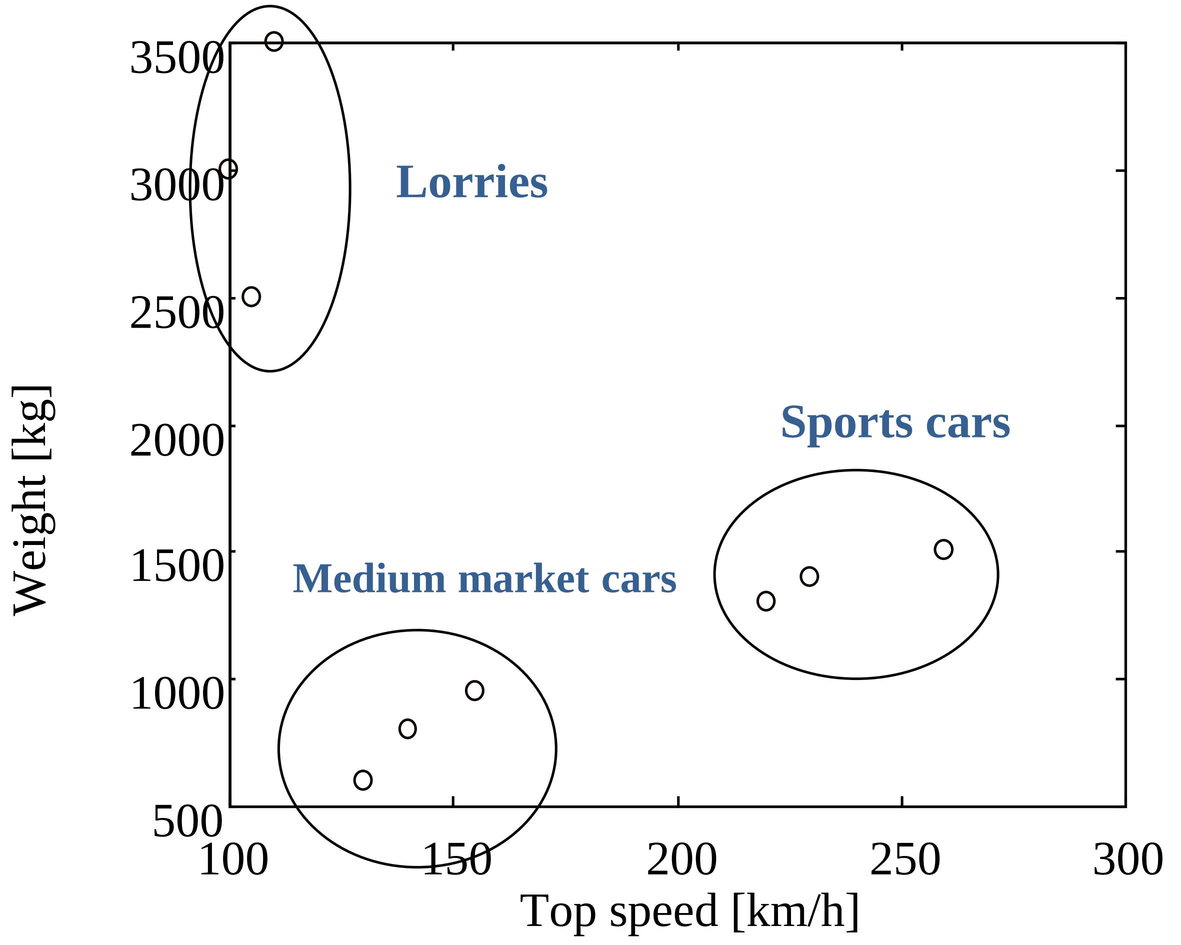
- Clustering example
  - 차량의 특성을 가지고 grouping 을 해 보자

Could see any group ?

Vehicle	Top speed km/h	Color	Air resistance	Weight Kg
V1	220	red	0.30	1300
V2	230	black	0.32	1400
V3	260	red	0.29	1500
V4	140	gray	0.35	800
V5	155	blue	0.33	950
V6	130	white	0.40	600
V7	100	black	0.50	3000
V8	105	red	0.60	2500
V9	110	gray	0.55	3500

Summary

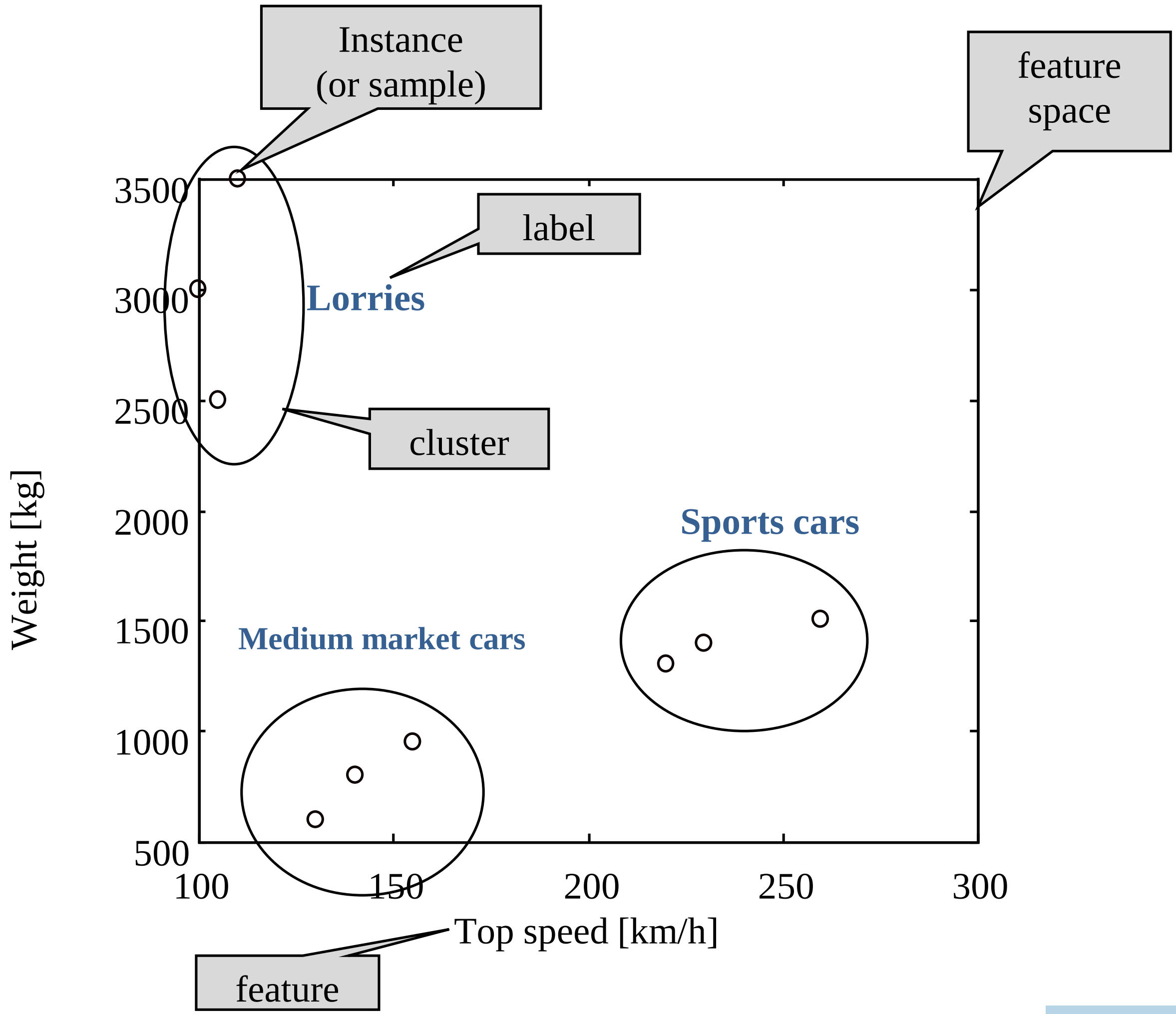
- Clustering example



Summary

- Clustering example

Terminology



K-means clusteinng

(Hard c-means (HCM))

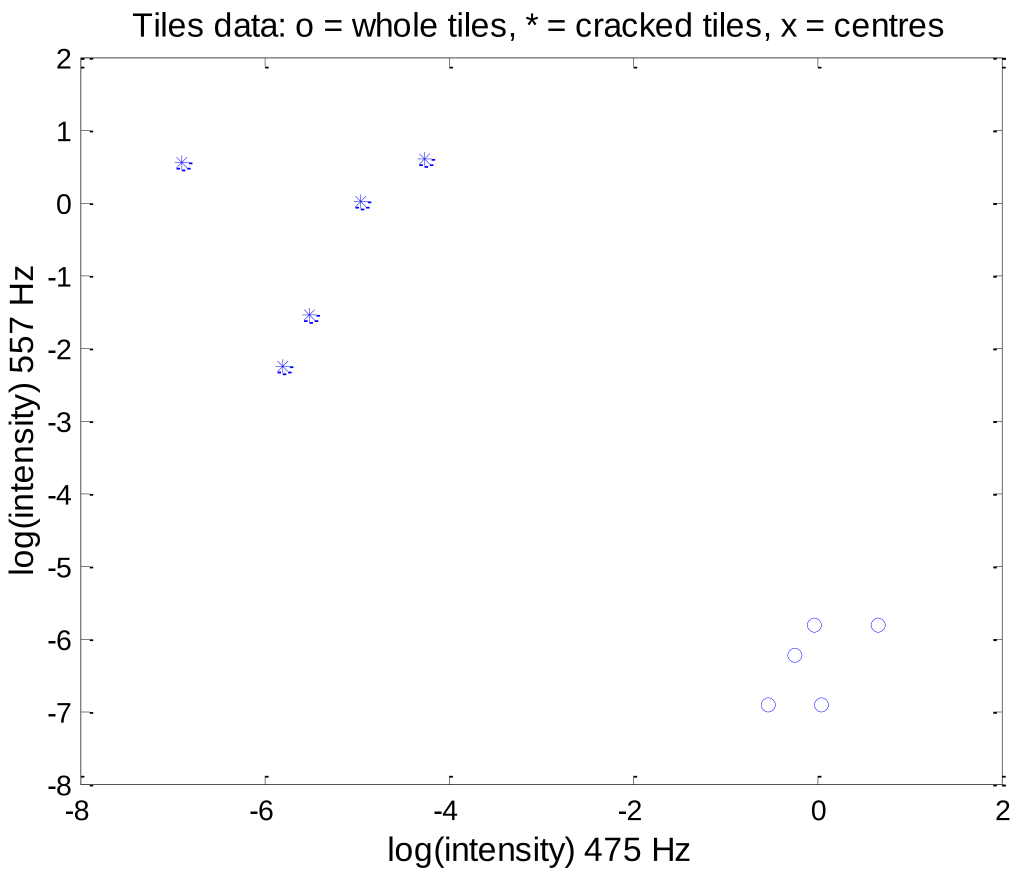
Cracked tiles example

475Hz	557Hz
0.958	0.003
1.043	0.001
1.907	0.003
0.780	0.002
0.579	0.001
0.003	0.105
0.001	1.748
0.014	1.839
0.007	1.021
0.004	0.214

Table 1: frequency intensities for ten tiles.

Tiles are made from clay moulded into the right shape, brushed, glazed, and baked. Unfortunately, the baking may produce invisible cracks. Operators can detect the cracks by hitting the tiles with a hammer, and in an automated system the response is recorded with a microphone, filtered, Fourier transformed, and normalised. A small set of data is given in TABLE 1 (adapted from MIT, 1997).

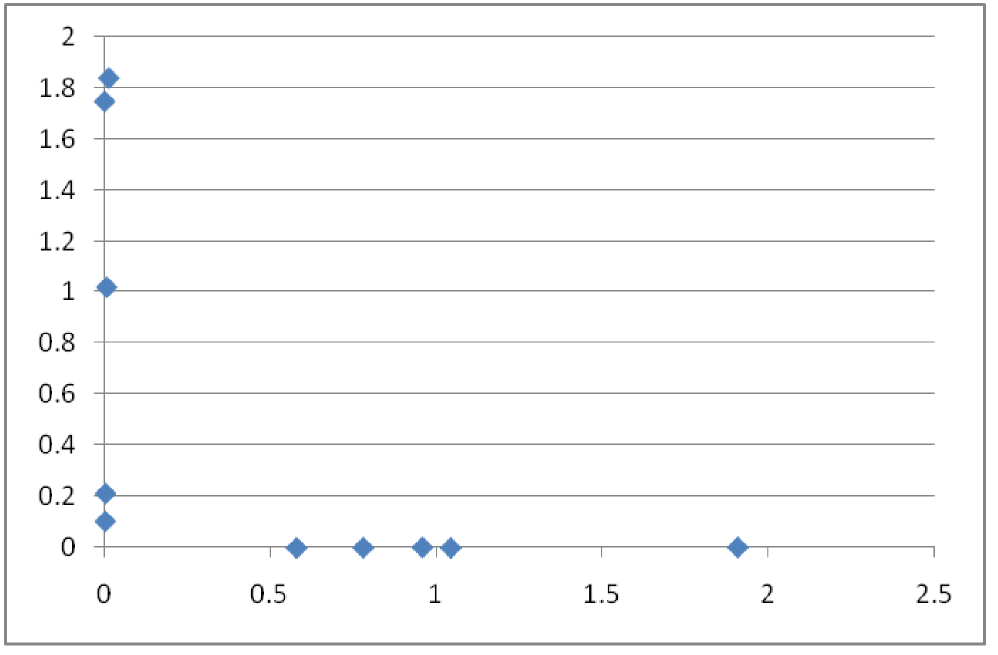
K-means



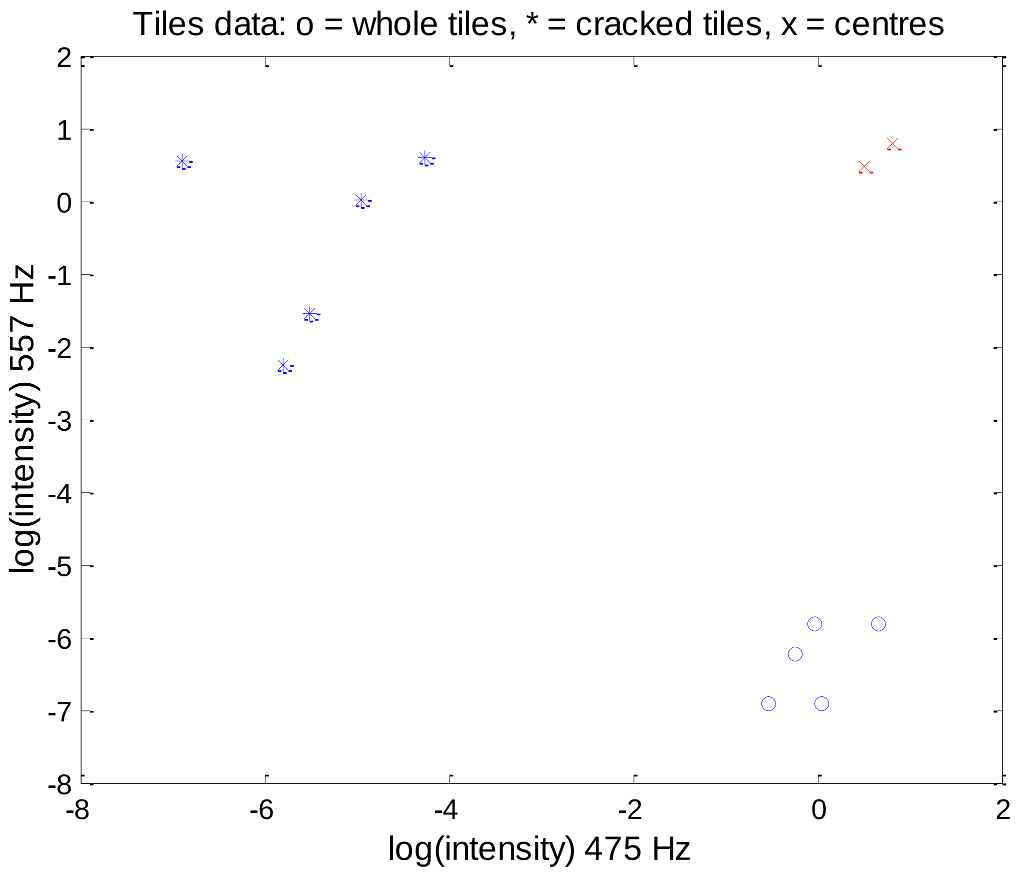
Plot of tiles by frequencies (logarithms). The whole tiles (o) seem well separated from the cracked tiles (*). The **objective** is to find the two clusters.

K-means

- Before logarithms

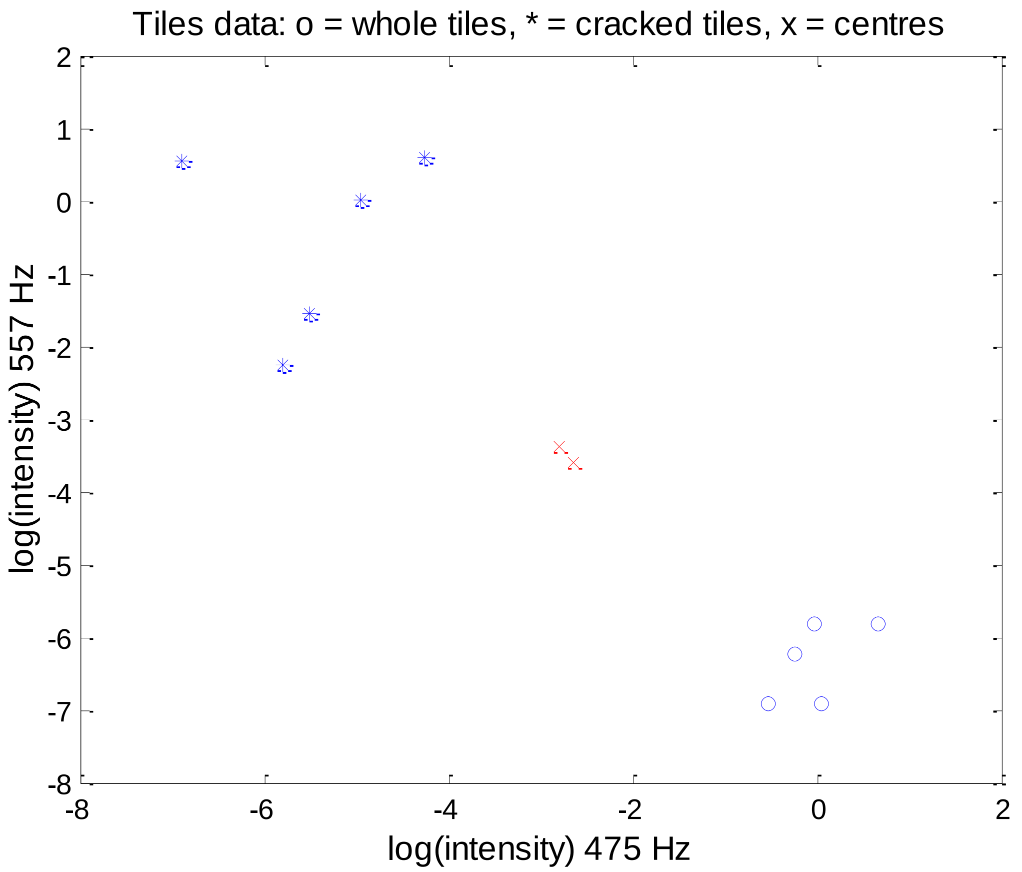


K-means



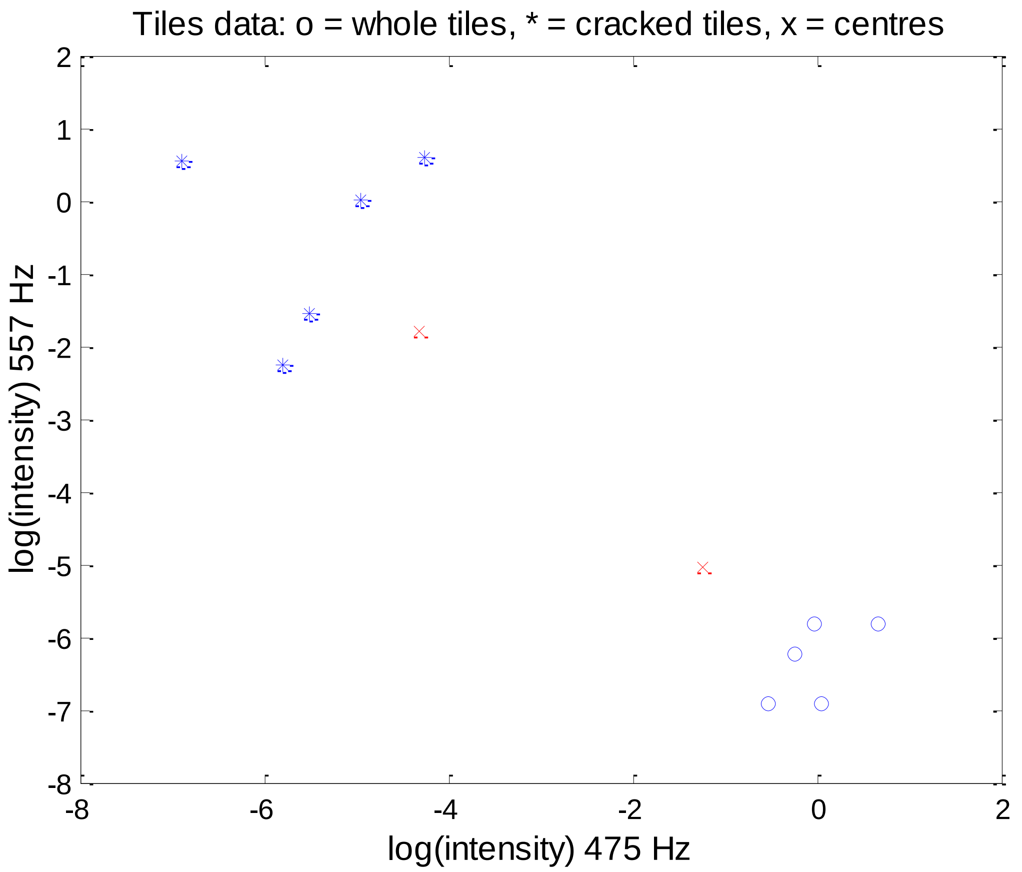
1. Place two cluster centres (x) at random.
2. Assign each data point (* and o) to the nearest cluster centre (x)

K-means



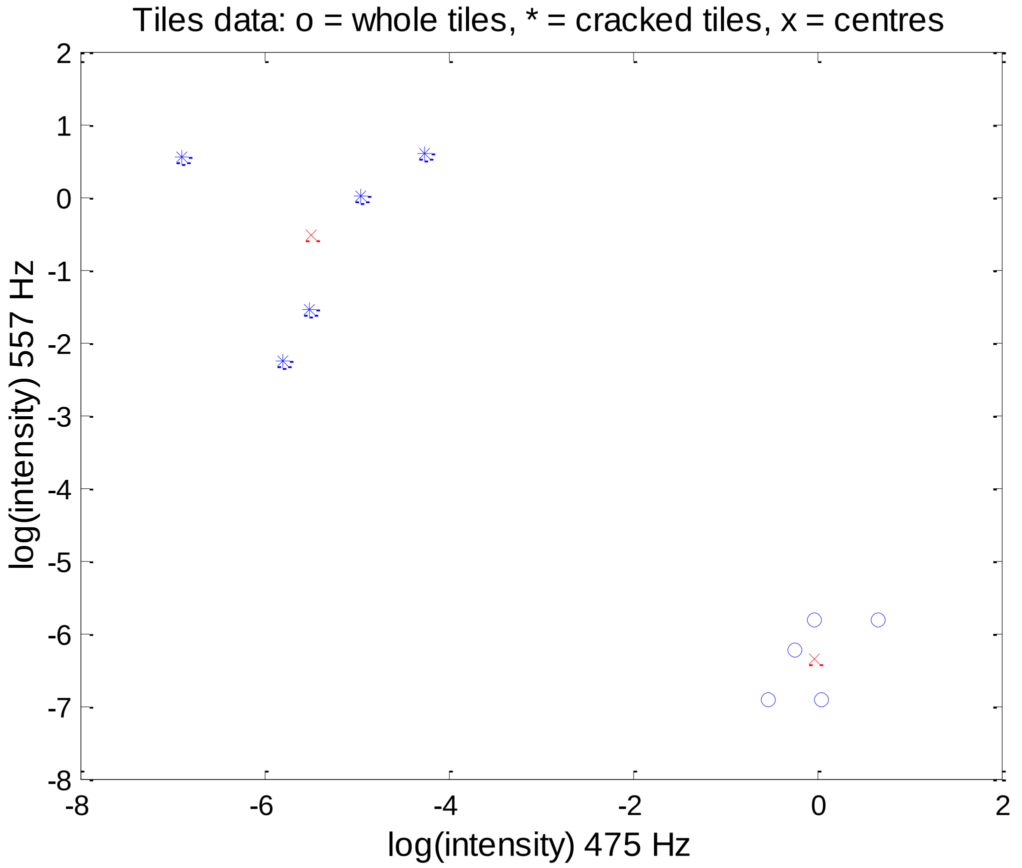
1. Compute the new centre of each class
2. Move the crosses (x)

K-means



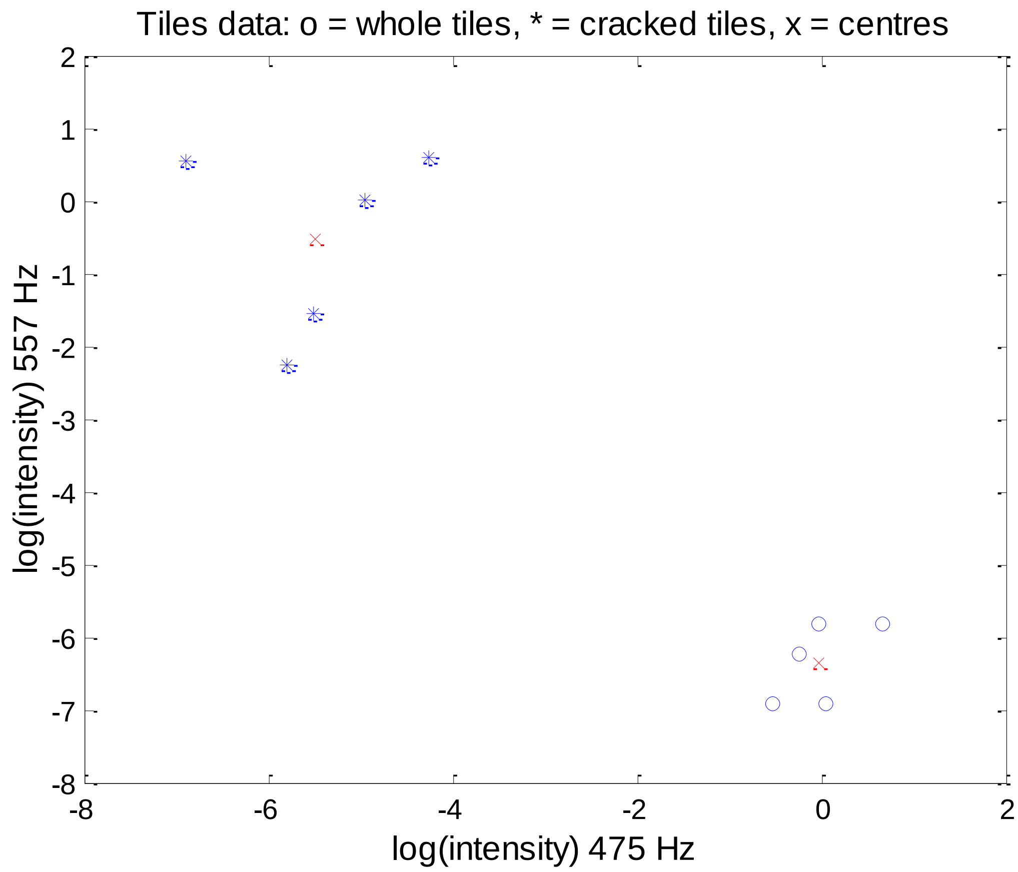
Iteration 2

K-means



Iteration 3

K-means



Iteration 4 (then stop, because no visible change)
Each data point belongs to the cluster defined by the nearest centre

K-means

475Hz	557Hz
0.958	0.003
1.043	0.001
1.907	0.003
0.780	0.002
0.579	0.001
0.003	0.105
0.001	1.748
0.014	1.839
0.007	1.021
0.004	0.214

M =

First cluster	Second cluster
0.0000	1.0000
0.0000	1.0000
0.0000	1.0000
0.0000	1.0000
0.0000	1.0000
1.0000	0.0000
1.0000	0.0000
1.0000	0.0000
1.0000	0.0000
1.0000	0.0000

The membership matrix M:

1. The last five data points (rows) belong to the first cluster (column)
2. The first five data points (rows) belong to the second cluster (column)

Membership matrix **M**

data point k

cluster centre i

cluster centre j

$$m_{ik} = \begin{cases} 1 \; if \; \left\| \mathbf{u}_k - \mathbf{c}_i \right\|^2 \le \left\| \mathbf{u}_k - \mathbf{c}_j \right\|^2 \\ 0 \; otherwise \end{cases}$$

distance

c-partition

- Property of clustering

All clusters C together fills the whole universe U

$$\bigcup_{i=1}^{c} C_i = U$$

Clusters do not overlap

$$C_i \cap C_j = \emptyset \quad for\ all\ i \neq j$$

A cluster C is never empty and it is smaller than the whole universe U

$$\emptyset \subset C_i \subset U \quad for\ all\ i$$

There must be at least 2 clusters in a c-partition and at most as many as the number of data points K

$$2 \leq c \leq K$$

Euclidean distance

p= $(p_1, p_2, p_3, \ldots, p_n)$, q = $(q_1, q_2, q_3, \ldots, q_n)$

Euclidean distance

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

scolar

[R 실습] : K-means

- Usage

```
kmeans(x, centers, iter.max = 10, nstart = 1,
algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
"MacQueen"))
```

- Argument
  - **x** : numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
  - **centers** : either the number of clusters, say *k*, or a set of initial (distinct) cluster centres.
  - **iter.max** : the maximum number of iterations allowed.
  - **nstart** : if centers is a number, how many random sets should be chosen? (20 또는 25 권장)
  - **algorithm** : character: may be abbreviated.

[R 실습] : K-means

- Return values
  - **cluster** : A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
  - **centers** : A matrix of cluster centres.
  - **withinss** : Vector of within-cluster sum of squares, one component per cluster.
  - **tot.withinss** : Total within-cluster sum of squares, i.e. sum(withinss).
  - **betweenss** : The between-cluster sum of squares, i.e. totss-tot.withinss.
  - **size** : The number of points in each cluster.

[R 실습] : K-means

```
require(graphics)
# a 2-dimensional example
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3),
           ncol = 2))
colnames(x) <- c("x", "y")
plot(x)
cl <- kmeans(x, 2)
cl # show clustering result
plot(x, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 8, cex=2)

kmeans(x,1)$withinss

# random starts do help here with too many clusters
cl <- kmeans(x, 5, nstart = 25)
plot(x, col = cl$cluster)
points(cl$centers, col = 1:5, pch = 8)
```

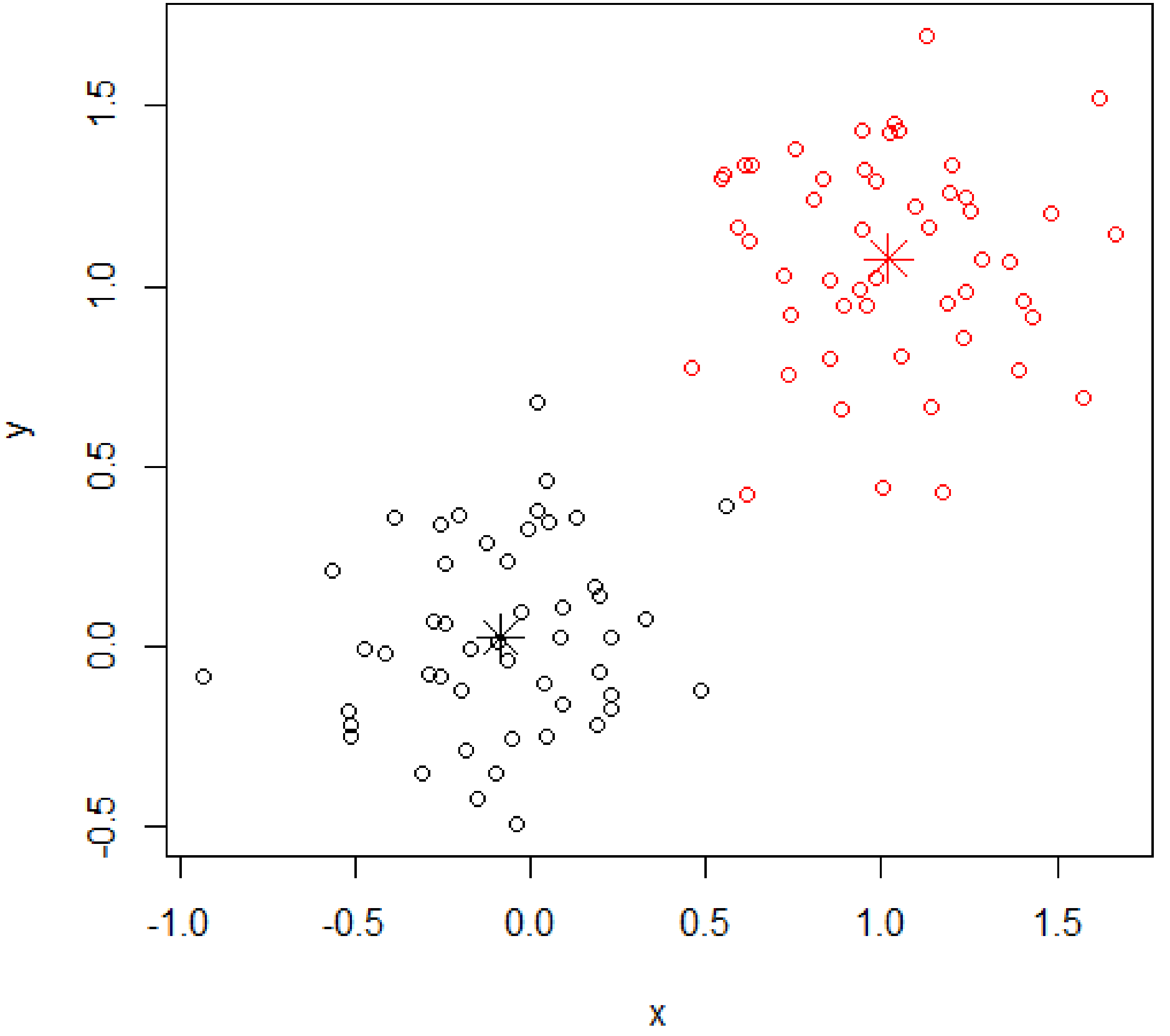
[R 실습] : K-means

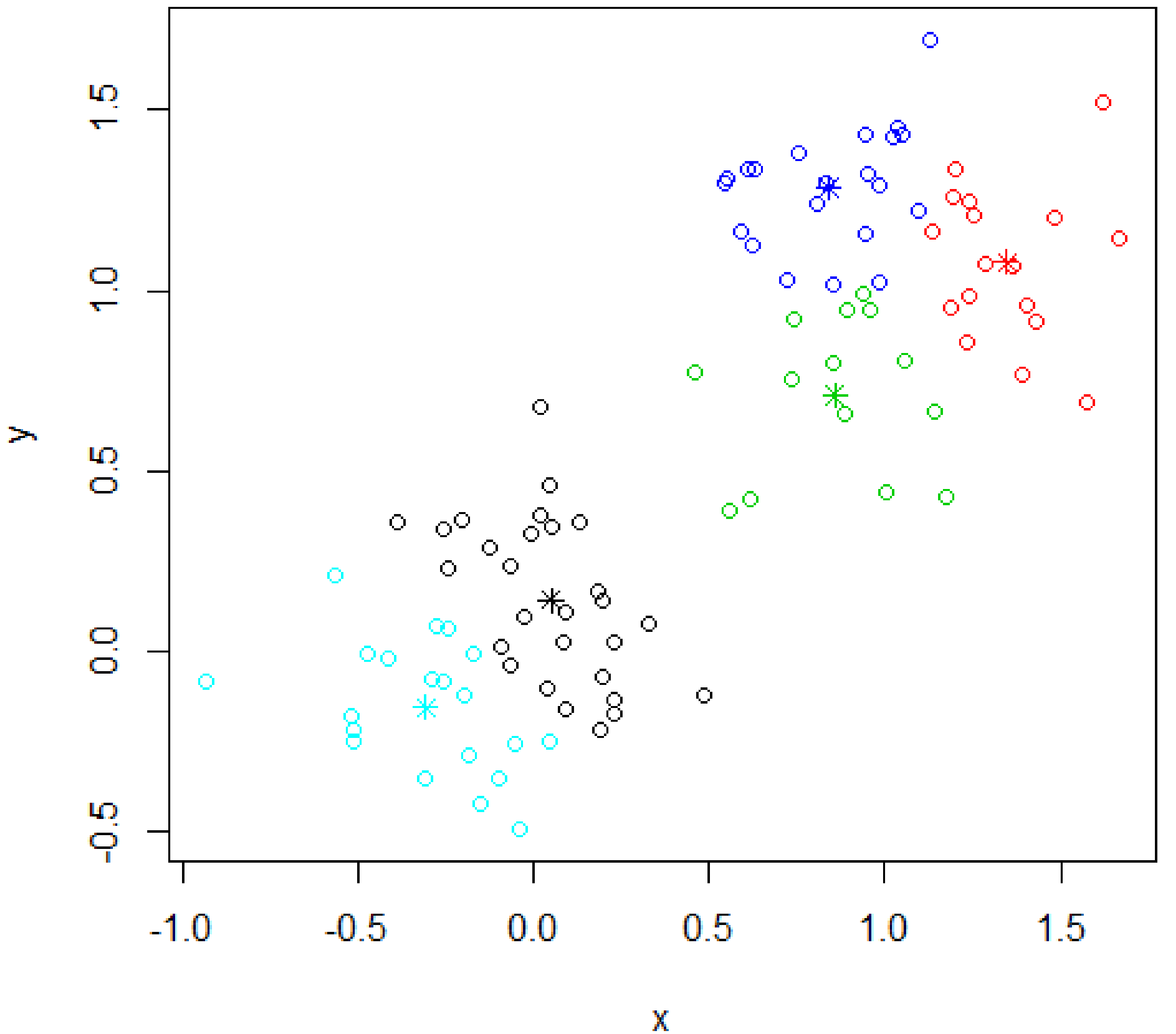
```
> cl
K-means clustering with 2 clusters of sizes 49, 51

Cluster means:
            x          y
1 -0.08673691 0.02745475
2  1.01929789 1.07596115

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [37] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [73] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 7.078592 8.616434
 (between_SS / total_SS =  78.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

[R 실습] : K-means

```
> cl$cluster
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [37] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [73] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> cl$centers
            x          y
1 -0.08673691 0.02745475
2  1.01929789 1.07596115
> cl$withinss
[1] 7.078592 8.616434
> cl$betweenss
[1] 58.04374
> cl$size
[1] 49 51
```

[R 실습] : K-means





1.5
1.0
0.5
0.0
-0.5
y
-1.0
-0.5
0.0
0.5
1.0
1.5
x

[R 실습] : K-means

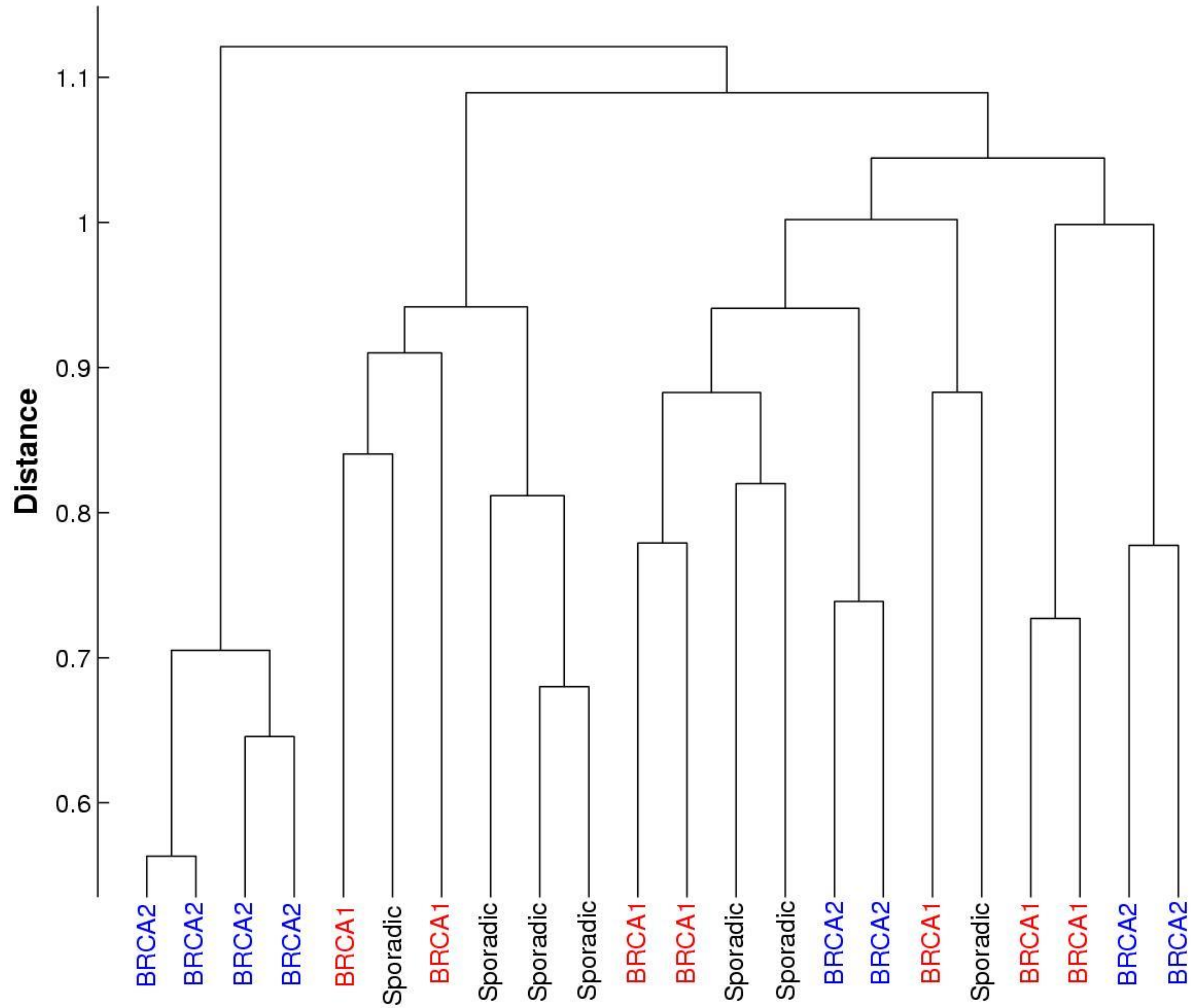
- 제공한 snsdata.csv 파일에 대해 kmeans clustering test 를 하시오
  - Remove NA rows
  - Collect data by 18 ≤ age ≤ 20
  - Change gender value : M → 1, F → 0
  - Set k = 5

gradyear	gender	age	friends	basketball	football	soccer	softball	volleyball	swimming	cheerl
2006	M	18.982	7	0	0	0	0	0	0	
2006	F	18.801	0	0	1	0	0	0	0	
2006	M	18.335	69	0	1	0	0	0	0	
2006	F	18.875	0	0	0	0	0	0	0	
2006	NA	18.995	10	0	0	0	0	0	0	
2006	F		142	0	0	0	0	0	0	
2006	F	18.93	72	0	0	0	0	0	0	
2006	M	18.322	17	0	0	0	1	0	0	
2006	F	19.055	52	0	0	0	0	0	0	
2006	F	18.708	39	0	0	0	0	0	0	

  - Try to find characteristics of each cluster (consider above 6 features)

Hierarchical clustering

Hierarchical Clustering



Hierarchical Clustering

- Given a set of N items to be clustered, and an N*N distance (or similarity) matrix, the basic process of hierarchical clustering (defined by S.C. Johnson in 1967) is this:

① Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.

② Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.

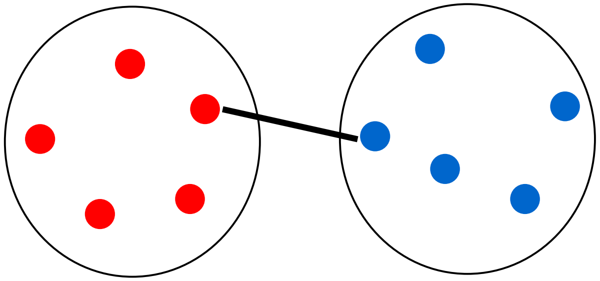
③ Compute distances (similarities) between the new cluster and each of the old clusters.

④ Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

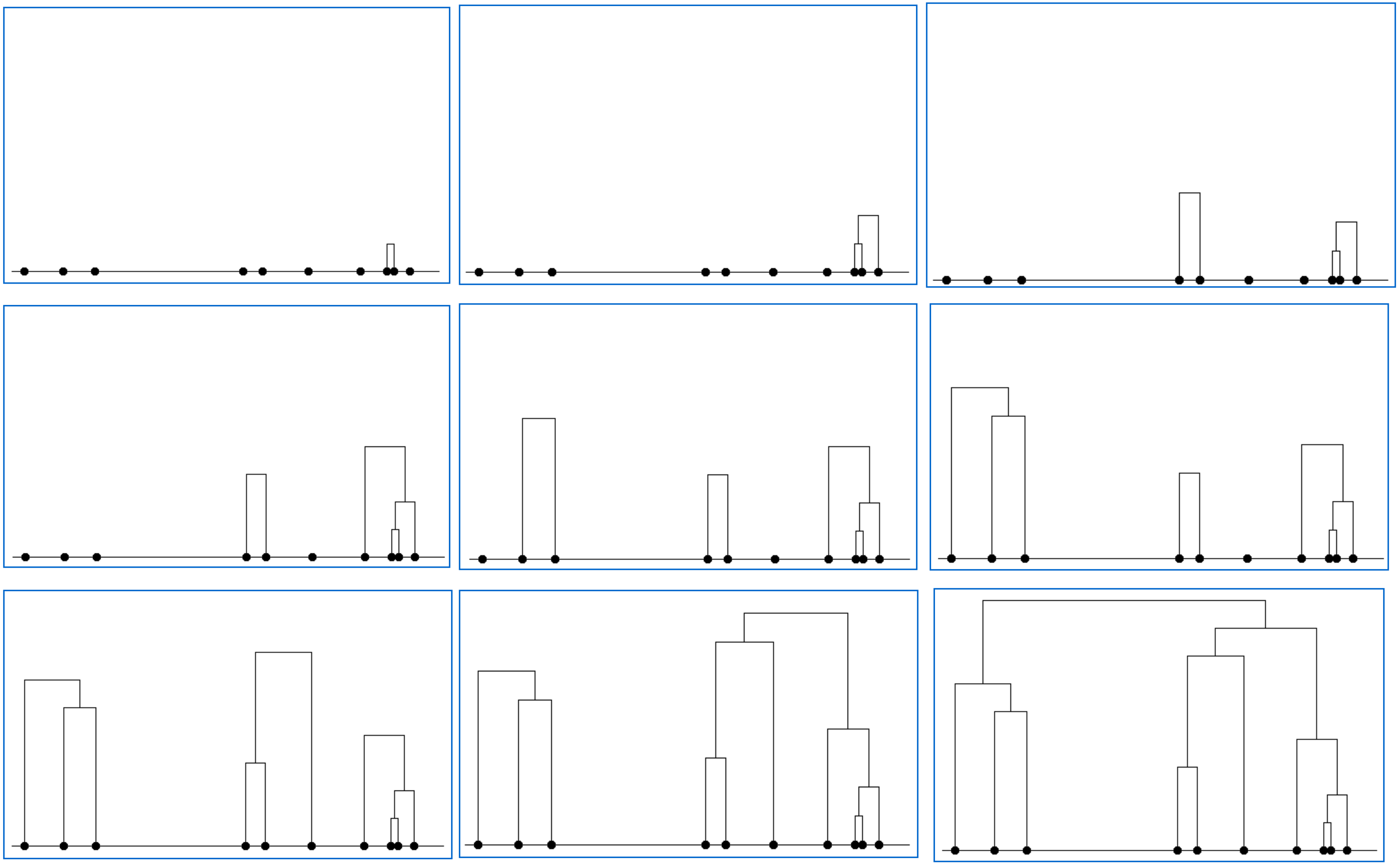
Hierarchical Clustering

- Step 3 can be done in different ways, which is what distinguishes single-linkage from complete-linkage and average-linkage clustering.

- In **single-linkage clustering** (also called the *connectedness* or *minimum* method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster.
- If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.

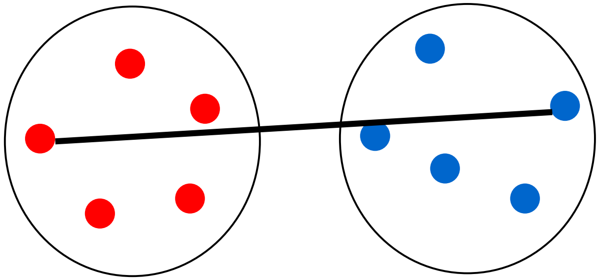


[Single-linkage]

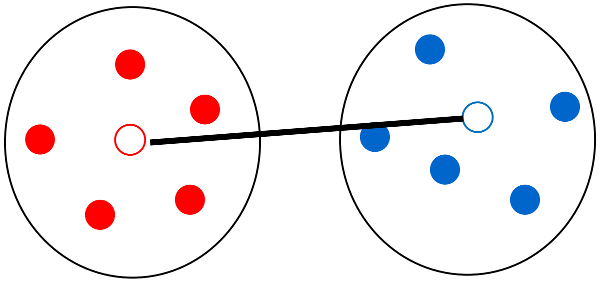


Hierarchical Clustering

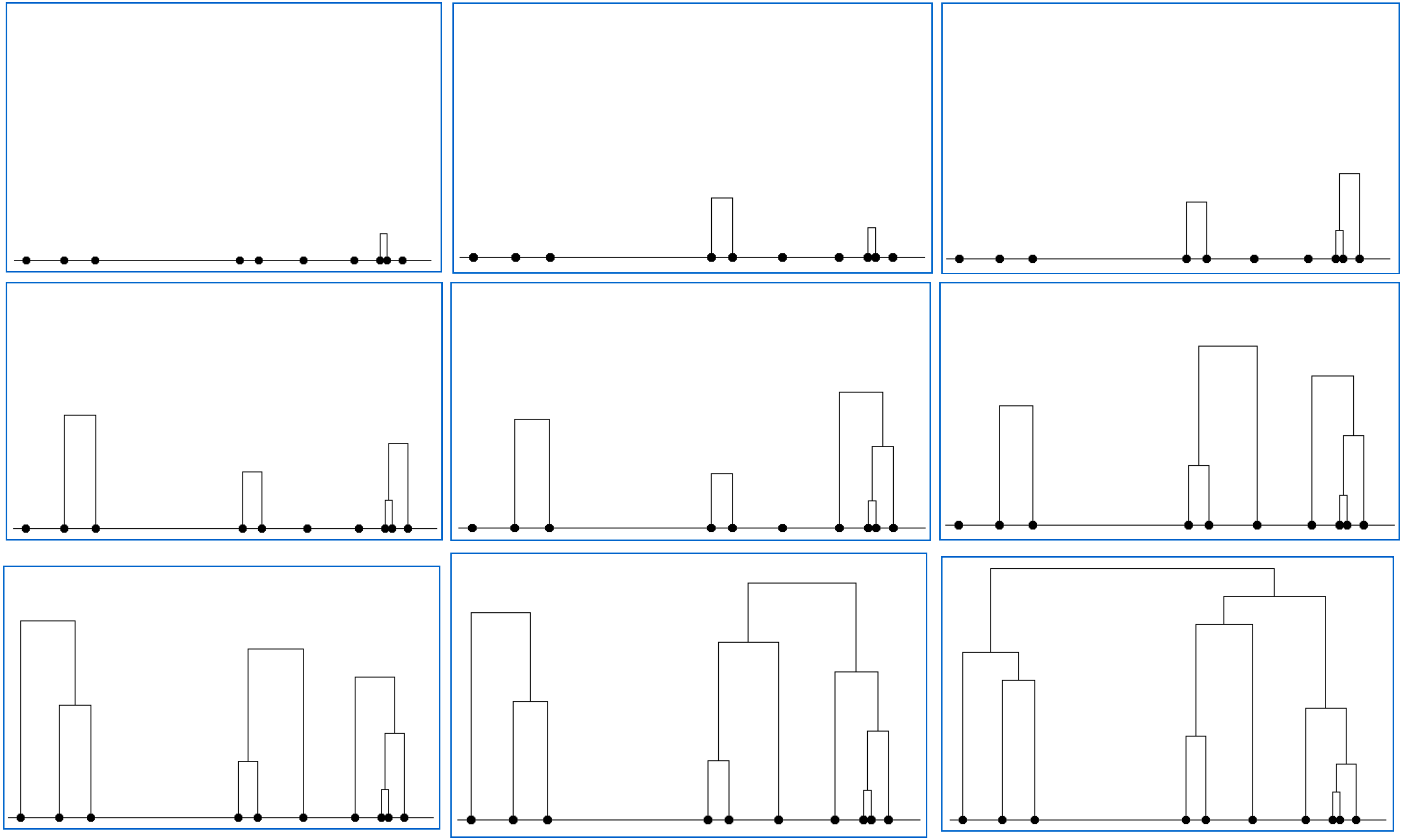
- In **complete-linkage clustering** (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster.



- In **average-linkage clustering**, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.



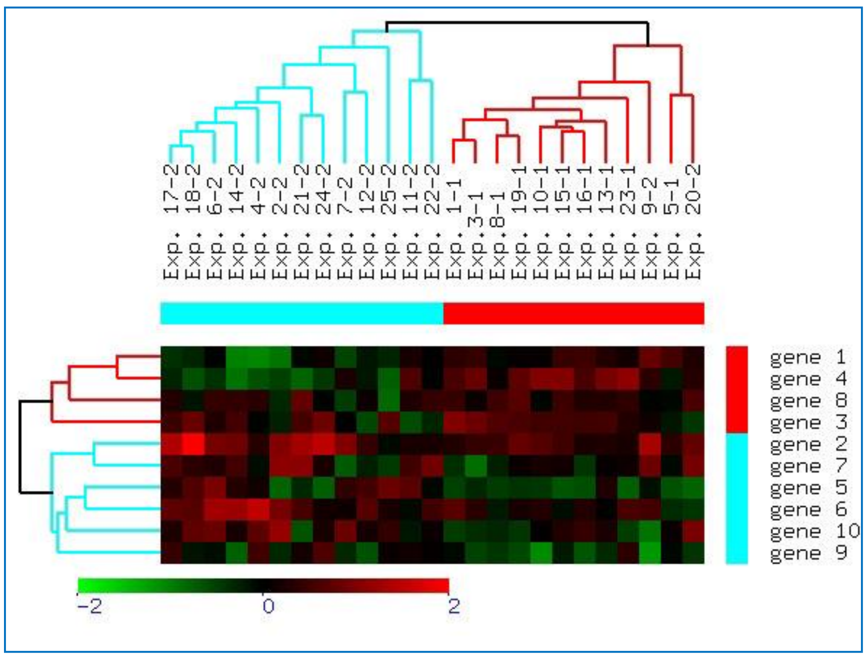
[Complete-linkage]



Hierarchical Clustering

- Problems
  - The main weaknesses of agglomerative clustering methods are:

  - they do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects;
  - they can never undo what was done previously.



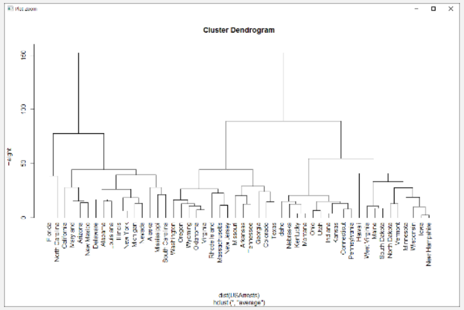
[R 실습]

- hclust (in stats library)
- Usage

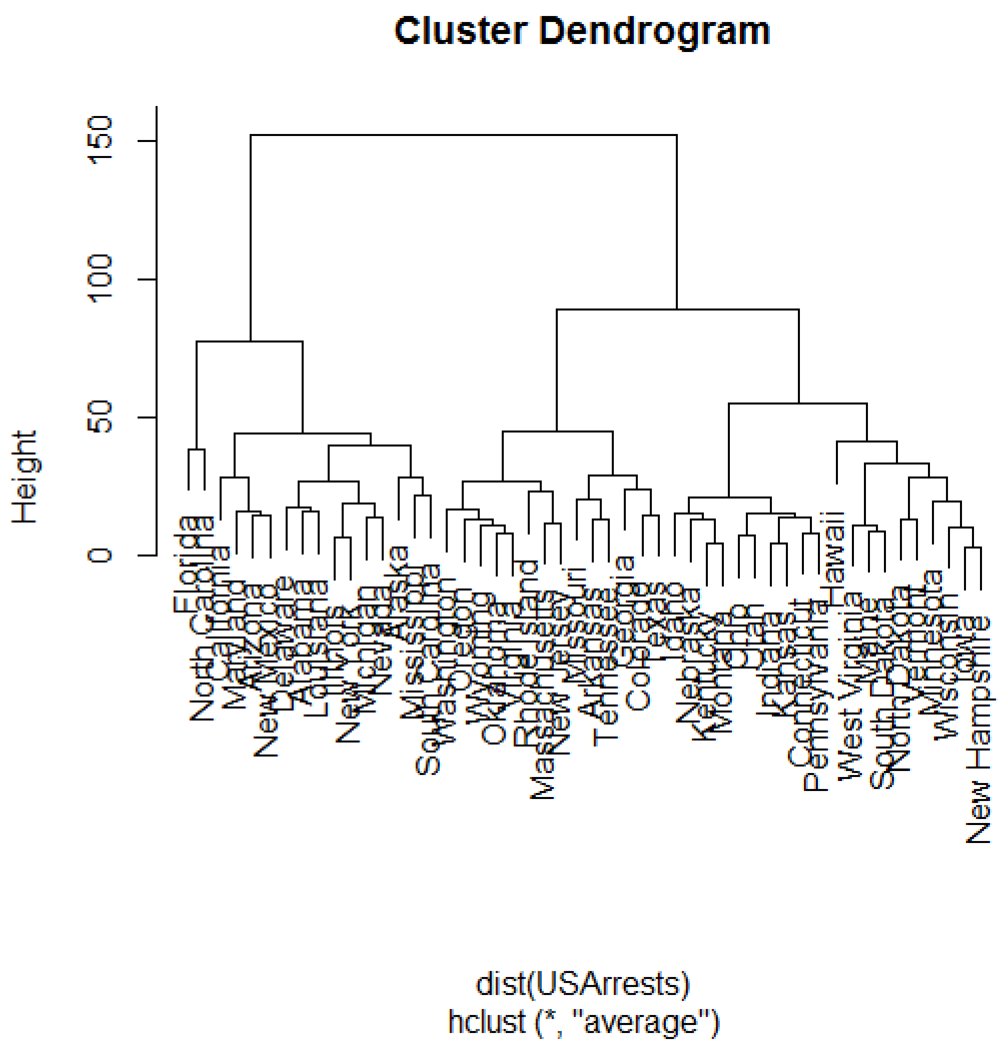
```
hclust(d, method = "complete", members=NULL)
```

- Argument
  - **d**: a dissimilarity structure as produced by dist.
  - **method** : the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single","complete", "average", "mcquitty", "median" or "centroid ".
  - **members**: NULL or a vector with length size of d. See the ‘Details’ section

```
require(stats)
require(graphics)
hc <- hclust(dist(USArrests), "ave")
plot(hc)
plot(hc, hang = -1)

## Do the same with centroid clustering and squared
## Euclidean distance,
## cut the tree into ten clusters and reconstruct
## the upper part of the
## tree from the cluster centers.
hc <- hclust(dist(USArrests)^2, "cen")
memb <- cutree(hc, k = 10)
memb
```

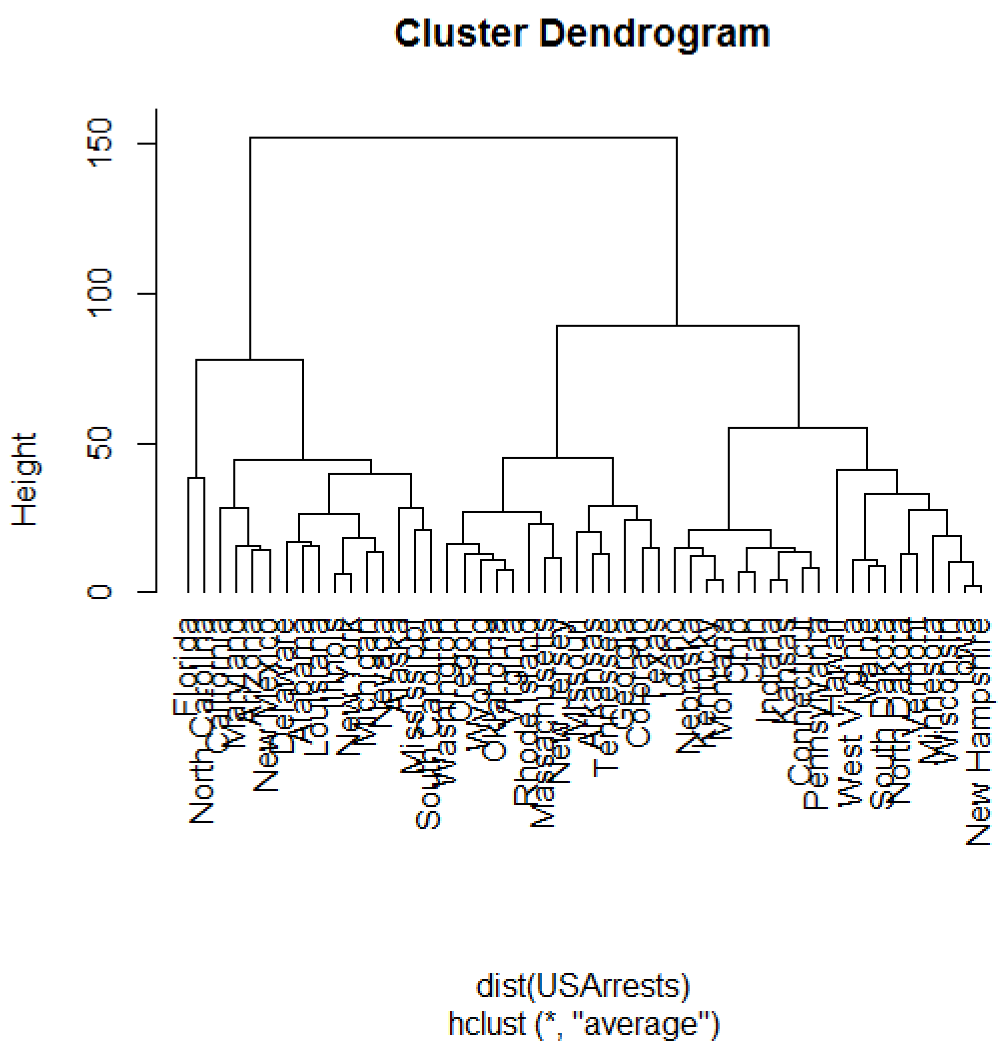


```
> plot(hc)
```



Cluster Dendrogram
Height
0
50
100
150
dist(USArrests)
hclust (*, "average")

```
> plot(hc, hang = -1)
```

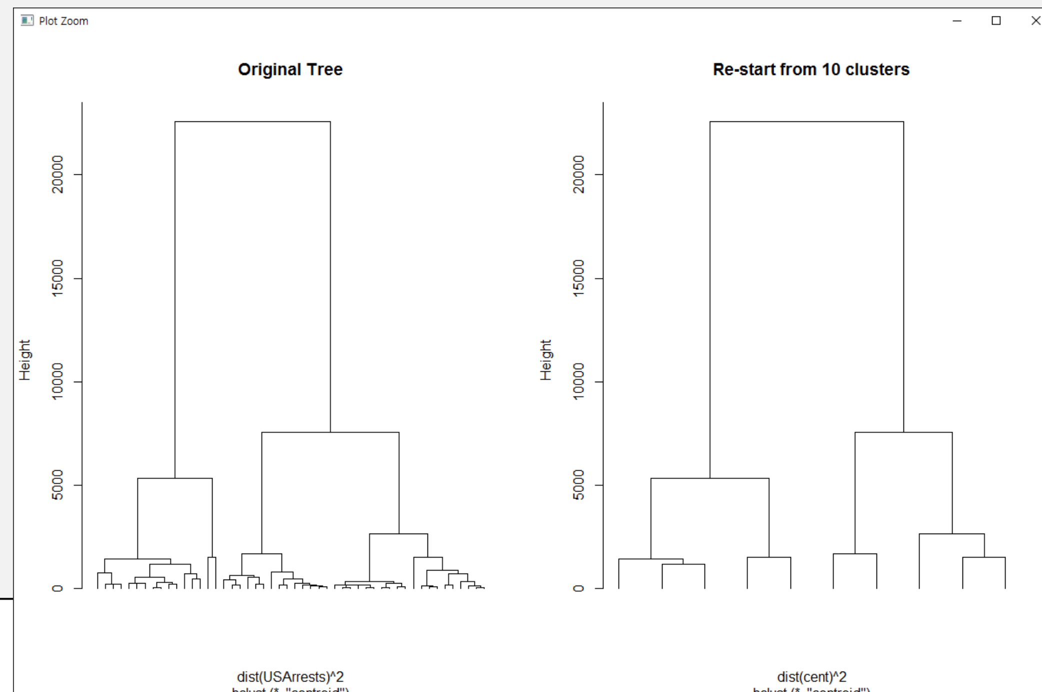


```
> memb
       Alabama         Alaska        Arizona       Arkansas
             1              2              3              4
    California       Colorado    Connecticut       Delaware
             3              4              5              1
       Florida        Georgia         Hawaii          Idaho
             6              4              7              5
      Illinois        Indiana           Iowa         Kansas
             1              5              8              5
      Kentucky      Louisiana          Maine       Maryland
             5              1              8              3
 Massachusetts       Michigan      Minnesota    Mississippi
             9              1              8              2
      Missouri        Montana       Nebraska         Nevada
             4              5              5              1
 New Hampshire     New Jersey     New Mexico       New York
             8              9              3              1
North Carolina   North Dakota           Ohio       Oklahoma
            10              8              5              9
        Oregon   Pennsylvania   Rhode Island South Carolina
             9              5              9              2
  South Dakota      Tennessee          Texas           Utah
             8              4              4              5
       Vermont       Virginia     Washington  West Virginia
             8              9              9              8
     Wisconsin        Wyoming
             8              9
```

```
## 10개 클러스터의 중심점 계산
cent <- NULL
for(k in 1:10){
    cent <- rbind(cent, colMeans(USArrests[memb == k, ,
                drop = FALSE]))
}
```

```
> cent
         Murder   Assault UrbanPop     Rape
 [1,] 11.471429 247.57143 74.28571 27.20000
 [2,] 13.500000 267.00000 46.66667 28.03333
 [3,]  9.950000 288.75000 77.00000 32.87500
 [4,] 11.500000 195.33333 66.16667 27.43333
 [5,]  5.590000 112.40000 65.60000 17.27000
 [6,] 15.400000 335.00000 80.00000 31.90000
 [7,]  5.300000  46.00000 83.00000 20.20000
 [8,]  2.688889  64.55556 50.66667 10.54444
 [9,]  5.750000 156.75000 74.00000 19.40000
[10,] 13.000000 337.00000 45.00000 16.10000
```

```
## 10개 클러스터를 재 클러스터링
hc1 <- hclust(dist(cent)^2, method = "cen",
       members = table(memb))
opar <- par(mfrow = c(1, 2))
plot(hc, labels = FALSE, hang = -1,
      main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1,
      main = "Re-start from 10 clusters")
par(opar)
```



[실습문제]

- UCI machine learning repository 에서 wine dataset 을 다운받아 hierarchical clustering 을 테스트 하시오.
  - http://archive.ics.uci.edu/ml/
  - 첫번째 컬럼은 class data 이니 클러스터링에서 제외

- 각 컬럼의 데이터 범위가 각기 달라서 거리 계산시 문제가 있음
  - 따라서 각 컬럼의 데이터 범위가 0 ~1 사이가 되도록 조정

- 1) **`plot(hc, hang = -1)`** 를 이용하여 그래프 작성
- 2) Tree 를 3개의 cluster 가 되도록 잘라서 그래프를 그린다
- 3) 3개로 클러스터링 된 결과를 실제 class 와 비교하여 일치도가 얼마나 되는지를 보이시오