Topic 9: Clustering (Unsupervised Learning)

Clustering is a broad class of unsupervised learning methods for finding "clusters" by labelling data (usually with integers) [James et al., 2013, Section 10.3]. Its primary purposes are (i) to group objects based on the **similarity in the features** and (ii) to reduce the information from an entire sample into information about smaller, manageable subgroups. A good clustering algorithm should produce high quality clusters with high intra-cluster (within-cluster) similarity and low inter-cluster (between-cluster) similarity for clustered data.

According to Yao [2008], clustering methods can be classified into

- Partitioning methods;
- Density-Based methods;
- Grid-Based methods:
- Hierarchical methods;
- Kernel and Spectral methods;
- Hidden Markov models: uses observed data to recover the sequence of states.

The texts for clustering are [Rencher, 2002, Chapter 14] and Jain and Dubes [1988].

Python's Scikit-learn and R support the following types of clustering algorithms.

Methods	Parameters	Usecase	Metric	R packages
K-Means	number of clusters	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points	stats:kmeans(), clusterR, cluster

Gaussian mixtures	many	Flat geometry, good for density estimation	Mahalanobis distances to centres	ClusterR, mixture
Affinity propagation	damping, sample preference	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbour graph)	clusterR, apcluster
Mean-shift	bandwidth	Many clusters, uneven cluster size, non-flat geometry	Distances between points	meanShiftR, LPCM::ms
Spectral clustering	number of clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)	kernlab, Spectrum
DBSCAN	neighbourhood size	Non-flat geometry, uneven cluster sizes	Distances between nearest points	dbscan::dbscan, fpc
OPTICS	minimum cluster membership	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points	dbscan::optics
AGNES	number of clusters or distance threshold, linkage type, distance	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance	stats:hclust(), cluster:agnes(), clust, fastcluster, flashClust
BIRCH	branching factor, threshold, optional global clusterer.	Large dataset, outlier removal, data reduction.		stream::DSC_BIRCH

Let us recall the definition of "dissimilarity" from Section 2.1 which is fundamental to hierarchical clustering.

A *dissimilarity (function)* $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a function which satisfies the following conditions: For any $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, $\boldsymbol{x}_k$,

(i) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$ and $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ iff $\boldsymbol{x}_i = \boldsymbol{x}_j$;

(ii) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$; and

A *distance function* is a dissimilarity function which satisfies

(iii) $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_k) + d(\boldsymbol{x}_k, \boldsymbol{x}_j)$ (triangle inequality).

9.1 Partitioning Methods

There are many partitioning methods such as k-Means clustering, k-Medoids clustering (partitions around medroids (PAM) in particular), k-Prototype clustering, CLARA, CLARANS, GMM (Gaussian mixture models), etc. We will only be exploring the simplest of the partitioning algorithms.

9.1.1 k-Means Clustering

The *k-means clustering* partitions a data set of n observations into K distinct, non-overlapping clusters (groups) $C_1, \cdots, C_K$ with the nearest mean, serving as a prototype for the cluster. It tries to find a "cluster" to optimise

$$\min_{C_1,\cdots,C_K} \left\{ \sum_{k=1}^{K} WSS_k \right\}, \quad WSS_k(C_k) = \sum_{\boldsymbol{x} \in C_k} d(\boldsymbol{x}, \overline{C_k})^2 \tag{9.1}$$

where WSS_k is the ***within sum of squares*** of the cluster k and $\overline{C_k}$ is the kth centroid.

In this section, we will only explore the simplest k-means clustering algorithm — Lloyd (1957) and Forgy (1965), other more algorithms will be ignored:

- Lloyd, S. P. (1957, 1982). Least squares quantization in PCM. Technical Note, Bell Laboratories. Published in 1982 in *IEEE Transactions on Information Theory*, 28, 128–137.
- Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, *21*, 768-769.
- Hartigan, J. A. and Wong, M. A. (1979): Algorithm AS 136: A K-means clustering algorithm. *Applied Statistics*, 28, 100–108. It considers both the first and second nearest clusters.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, eds L. M. Le Cam & J. Neyman, 1, pp. 281–297. Berkeley, CA: University of California Press. More complex that Lloyd-Forgy because it updates the centroids when it is assigning labels.
- Elkan, Charles (2003). Using the Triangle Inequality to Accelerate K-Means, AAAI Press.

The **Lloyd-Forgy algorithm** is roughly shown below [James et al., 2013, Chapter 10]:

1. Randomly pick K observations as the initial centroid(s) $\boldsymbol{c}_k$.
2. Iterate until the centroid(s) stop changing:
   (a) Assign each observation $\boldsymbol{x}_i$ to the cluster with the nearest centroid $\boldsymbol{c}_j$, i.e. partition the observations according to the Voronoi diagram generated by the centroid:
   $$C_k^{(t)} = \Big\{ \boldsymbol{x}_i : \forall j, 1 \le j \le K \big(d(\boldsymbol{x}_i, \boldsymbol{c}_k^{(t)}) \le d(\boldsymbol{x}_i, \boldsymbol{c}_j^{(t)}) \big), \quad 1 \le i \le n \Big\},$$
   where each $\boldsymbol{x}_i$ is assigned to exactly one $C_k^{(t)}$, even if it could be assigned to two or more of them.
   (b) Recalculate centroids for $\boldsymbol{x}_i$ assigned to each cluster: $\boldsymbol{c}_k^{(t+1)} = \dfrac{1}{|C_k^{(t)}|} \displaystyle\sum_{x_i \in C_k^{(t)}} \boldsymbol{x}_i$.

The implementation of k-means clustering in R and Python are listed below.

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

# Python
sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10,
  max_iter=300, tol=0.0001, verbose=0)
```

For Python, '`k-means++`' is the method for initialisation which selects initial cluster centres for k-mean clustering in a smart way to speed up convergence. An alternative is 'random', which chooses n_clusters observations (rows) at random from data for the initial centroids; and an `ndarray` in the shape of (n_clusters, n_features) which gives the initial centres. The `algorithm` is `lloyd` and `elkan`.

The return values of R's kmeans are:

- cluster: A vector of integers (from 1 to k) indicating the cluster to which each point is allocated.
- centers: A matrix of cluster centres.
- totss: The total sum of squares, i.e. sum of squares when there is only one cluster.
- withinss: Vector of within-cluster sum of squares, WSS_k in (9.1).
- 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.
- iter: The number of (outer) iterations.
- ifault: integer: indicator of a possible algorithm problem — for experts.

Example 9.1.1 (May 2022 Semester Final Exam, Q3(a))**.** Given the three-dimensional data in Table 3.1.

	x_1	x_2	x_3
A	2.6	3	4
B	1.4	4	5
C	2.5	2	2
D	1.7	2	5
E	2.7	3	4
F	2.4	4	4

Table 3.1: Three-dimensional data for clustering.

Perform k-means clustering algorithm (using the Euclidean distance) on the data from Table 3.1 with A and F as the initial centres until two clusters are found. Write down the stable cluster centres. You may round the numbers in your calculations to 4 decimal places.

(12 marks)

We can obtain **one** k-means clustering using the Lloyd-Forgy algorithm.

```
X = read.csv(text="
x1,x2,x3
2.6,3,4
1.4,4,5
2.5,2,2
1.7,2,5
2.7,3,4
2.4,4,4
", row.names=LETTERS[1:6])

ini = X[c("A","F"),]
km = kmeans(X, ini, algorithm="Lloyd")  # km = kmeans(X, ini, trace=T)
print(km)
```

The output of the Lloyd-Forgy algorithm (note that the Hartigan-Wong algorithm gives us a different k-means clustering).

```
K-means clustering with 2 clusters of sizes 4, 2
```

```
Cluster means:
     x1  x2   x3
1 2.375 2.5 3.75            <- km$centers
2 1.900 4.0 4.50

Clustering vector:
A B C D E F
1 2 1 1 1 2                 <- km$cluster

Within cluster sum of squares by cluster:
[1] 6.3775 1.0000           <- km$withinss
 (between_SS / total_SS =  35.4 %)

Available components:

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

Solution: Given the initial cluster centres 1 and 2:

$$A(2.6, 3, 4), \quad F(2.4, 4, 4)$$

Step 1 : Update table based on distance to cluster centres

x1	x2	x3	dist.1	dist.2	clust.centre
2.6	3	4	0	1.0198	1
1.4	4	5	1.8547	1.4142	2
2.5	2	2	2.2383	2.8302	1
1.7	2	5	1.6763	2.3431	1
2.7	3	4	0.1	1.044	1
2.4	4	4	1.0198	0	2

..... [7 marks]

The new cluster centres 1 and 2 are

$$\begin{aligned} Centre_1 &= (2.375, 2.5, 3.75), \\ Centre_2 &= (1.9, 4, 4.5) \end{aligned}$$ [1 mark]

Step 2 : Update table based on distance to cluster centres

x1	x2	x3	dist.1	dist.2	clust.centre
2.6	3	4	0.6026	1.3191	1
1.4	4	5	2.1825	0.7071	2
2.5	2	2	1.8243	3.2573	1
1.7	2	5	1.506	2.0712	1
2.7	3	4	0.6466	1.3748	1
2.4	4	4	1.5209	0.7071	2

..... [3 marks]

The stable cluster centres are

$$\begin{aligned} Centre_1 &= (2.375, 2.5, 3.75), \\ Centre_2 &= (1.9, 4, 4.5) \end{aligned}$$ [1 mark]

Example 9.1.2 (May 2022 Semester Final Exam, Q4(c))**.** Given the three-dimensional points

in Table 4.2.

Obs.	x_1	x_2	x_3
P_1	3.3	4.4	2.5
P_2	2.4	3.1	2.1
P_3	0.1	1.9	1.1
P_4	0.3	2.4	1.5
P_5	0.6	1.1	1.1
P_6	2.9	0.1	0.1
P_7	4.3	6.4	5.5
P_8	3.4	5.1	5.1
P_9	1.1	3.9	4.1

Table 4.2: Three-dimensional points.

Use the k-means clustering method with **Manhattan distance** to cluster the given points into $k = 3$ clusters by using P_6, P_3, P_5 as the initial clusters, find the **stable cluster centres/centroids**. (8 marks)

Solution: Given the initial centres:

$$Centre_1 = (2.9, 0.1, 0.1), \quad Centre_2 = (0.1, 1.9, 1.1), \quad Centre_3 = (0.6, 1.1, 1.1)$$

Step 1 : Update table based on distance to cluster centres

Obs.	x_1	x_2	x_3	dist.1	dist.2	dist.3	label
P_1	3.3	4.4	2.5	7.1	7.1	7.4	1
P_2	2.4	3.1	2.1	5.5	4.5	4.8	2
P_3	0.1	1.9	1.1	5.6	0	1.3	2
P_4	0.3	2.4	1.5	6.3	1.1	2	2
P_5	0.6	1.1	1.1	4.3	1.3	0	3
P_6	2.9	0.1	0.1	0	5.6	4.3	1
P_7	4.3	6.4	5.5	13.1	13.1	13.4	1
P_8	3.4	5.1	5.1	10.5	10.5	10.8	1
P_9	1.1	3.9	4.1	9.6	6	6.3	2

..... [3 marks]

The new cluster centres are ..... [1 mark]

$$Centre_1 = (3.475, 4, 3.3), \quad Centre_2 = (0.975, 2.825, 2.2), \quad Centre_3 = (0.6, 1.1, 1.1)$$

Step 2 and Step 3 : Update table based on distance to cluster centres

x_1	x_2	x_3	dist.1^2	dist.2^2	dist.3^2	label^2	dist.1^3	dist.2^3	dist.3^3	label^3
3.3	4.4	2.5	1.375	4.2	7.4	1	3.1333	3.3667	7.2	1
2.4	3.1	2.1	3.175	1.8	4.8	2	5.7333	1.6333	4.6	2
0.1	1.9	1.1	7.675	2.9	1.3	3	10.2333	3.8667	2.3	3
0.3	2.4	1.5	6.575	1.8	2	2	9.1333	2.7667	3	2
0.6	1.1	1.1	7.975	3.2	0	3	10.5333	4.1667	1	3
2.9	0.1	0.1	7.675	6.75	4.3	3	10.2333	7.1333	3.3	3
4.3	6.4	5.5	5.425	10.2	13.4	1	2.8667	9.2333	13.2	1
3.4	5.1	5.1	2.975	7.6	10.8	1	1.2	6.6333	10.6	1
1.1	3.9	4.1	3.275	3.1	6.3	2	4.2333	2.4667	6.3	2

.....[2+1=3 marks]

The cluster centres from Step 2 and the stable cluster centres are

$Centre_1 = (3.6667, 5.3, 4.3667)$, $Centre_2 = (1.2667, 3.13333, 2.5667)$,
$Centre_3 = (1.2, 1.03333, 0.7667)$ ..... [1 mark]

Example 9.1.3. You are given the following observations with two variables, x_1 and x_2.

Obs	x_1	x_2
A	1	1
B	1	2
C	2	1
D	2	2
E	8	8
F	8	9
G	9	8
H	9	9

Perform k-means clustering to group the observations into two groups using Euclidean distance. Given that the random initial centroids be D and A.

Solution:

Initial Centroids: $\boldsymbol{c}_1^{(0)} = (2, 2)$, $\boldsymbol{c}_2^{(0)} = (1, 1)$

Step 1:

Obs	x	y	distance from $\boldsymbol{c}_1$	distance from $\boldsymbol{c}_2$	cluster
A	1	1	1.414214	0.000000	2
B	1	2	1.000000	1.000000	1
C	2	1	1.000000	1.000000	1
D	2	2	0.000000	1.414214	1
E	8	8	8.485281	9.899495	1
F	8	9	9.219544	10.630146	1
G	9	8	9.219544	10.630146	1
H	9	9	9.899495	11.313708	1

Update Centroids: $\boldsymbol{c}_1^{(1)} = \frac{(1,2)+...+(9,9)}{7} = (5.571429, 5.571429)$, $\boldsymbol{c}_2^{(1)} = (1, 1)$

Step 2:

Obs	x	y	distance from $\boldsymbol{c}_1$	distance from $\boldsymbol{c}_2$	cluster
A	1	1	6.464976	0.000000	2
B	1	2	5.801126	1.000000	2
C	2	1	5.801126	1.000000	2
D	2	2	5.050763	1.414214	2
E	8	8	3.434519	9.899495	1
F	8	9	4.201555	10.630146	1
G	9	8	4.201555	10.630146	1
H	9	9	4.848732	11.313708	1

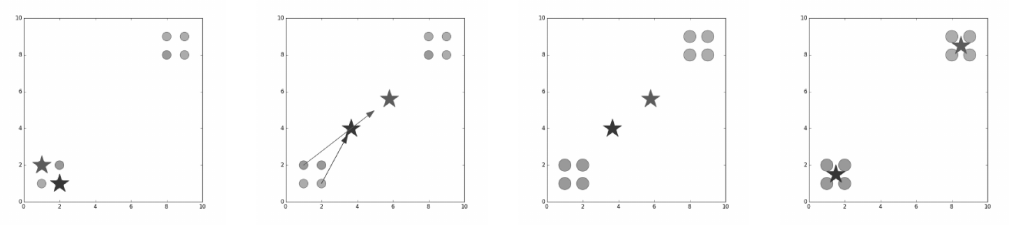
Update Centroids: $\boldsymbol{c}_1^{(1)} = \frac{(8,8)+...+(9,9)}{4} = (8.5, 8.5)$, $\boldsymbol{c}_2^{(1)} = \frac{(1,1)+...+(2,2)}{4} = (1.5, 1.5)$

Step 3:

Obs	x	y	distance from $\boldsymbol{c}_1$	distance from $\boldsymbol{c}_2$	cluster
A	1	1	10.6066017	0.7071068	2
B	1	2	9.9247166	0.7071068	2
C	2	1	9.9247166	0.7071068	2
D	2	2	9.1923882	0.7071068	2
E	8	8	0.7071068	9.1923882	1
F	8	9	0.7071068	9.9247166	1
G	9	8	0.7071068	9.9247166	1
H	9	9	0.7071068	10.6066017	1

No further changes in the closest cluster labels. Stable/Final Centroids: $\boldsymbol{c}_1^{(1)} = (8.5, 8.5)$, $\boldsymbol{c}_2^{(1)} = (1.5, 1.5)$

The visual presentation of the update of centroids for initial centroids B and C.



Initial Centroids Step 1 Step 2 Step 3

Example 9.1.4 (Feature Scaling for Data with Large Variance)**.** Given the following observations with two variables, x_1 and x_2.

Obs	x_1	x_2
A	0	0
B	0	2
C	20	0
D	20	2
E	80	8
F	80	10
G	100	8
H	100	10

(a) Perform k-means clustering to group the observations into two groups using Euclidean distance. Note that the range for x_1 is [0,100] and range for x_2 is [0,10]. Assume that the random initial centroids are B and E.

Solution:

Initial Centroids: $\boldsymbol{c}_1^{(0)} = (0, 2)$, $\boldsymbol{c}_2^{(0)} = (80, 8)$

Step 1:

	x_1	x_2	Dist1	Dist2	cluster
A	0	0	2.00000	80.39900	1
B	0	2	0.00000	80.22468	1
C	20	0	20.09975	60.53098	1
D	20	2	20.00000	60.29925	1
E	80	8	80.22468	0.00000	2
F	80	10	80.39900	2.00000	2
G	100	8	100.17984	20.00000	2
H	100	10	100.31949	20.09975	2

Update Centroids: $\boldsymbol{c}_1^{(1)} = \frac{(0,0)+\ldots+(20,2)}{4} = (10, 1)$, $\boldsymbol{c}_2^{(1)} = \frac{(80,8)+\ldots+(100,10)}{4} = (90, 9)$

Step 2:

	x_1	x_2	Dist1	Dist2	cluster
A	0	0	10.04988	90.44888	1
B	0	2	10.04988	90.27181	1
C	20	0	10.04988	70.57620	1
D	20	2	10.04988	70.34913	1
E	80	8	70.34913	10.04988	2
F	80	10	70.57620	10.04988	2
G	100	8	90.27181	10.04988	2
H	100	10	90.44888	10.04988	2

Stable Centroids: $\boldsymbol{c}_1^{(1)} = (10, 1)$, $\boldsymbol{c}_2^{(1)} = (90, 9)$

(b) Rescale both variables x_1 and x_2 using **standardisation** and perform k-means clustering.

Solution: First calculate the mean $\overline{X}_1 = \dfrac{0 + ... + 90}{14} = 50$; $\overline{X}_2 = \dfrac{0 + ... + 3}{14} = 5$ and then the (sample) standard deviation

$$(s_X)_1 = \sqrt{\frac{(0-50)^2 + \cdots + (90-50)^2}{14-1}} = \sqrt{1942.857143} = 37.00312$$
$$(s_X)_2 = \sqrt{\frac{(0-5)^2 + \cdots + (3-5)^2}{14-1}} = 3.86304$$

After the standardisation, if the k-mean clustering ($k = 2$) gives the cluster centres $\boldsymbol{c}_1 = (-0.8107425, -0.7765905)$ and $\boldsymbol{c}_2 = (0.8107425, 0.7765905)$, then

Obs	$\tilde{x}_1$	$\tilde{x}_2$	$d(\tilde{x}_1, \boldsymbol{c}_1)$	$d(\tilde{x}_1, \boldsymbol{c}_2)$	cluster
A	-1.3512375	-1.2943175	0.7484491	2.9937964	1
B	-1.3512375	-0.7765905	0.5404950	2.6620534	1
C	-0.8107425	-1.2943175	0.5177270	2.6301850	1
D	-0.8107425	-0.7765905	0.0000000	2.2453473	1
E	0.8107425	0.7765905	2.2453473	0.0000000	2
F	0.8107425	1.2943175	2.6301850	0.5177270	2
G	1.3512375	0.7765905	2.6620534	0.5404950	2
H	1.3512375	1.2943175	2.9937964	0.7484491	2
I	-1.0809900	0.5177270	1.3222297	1.9093617	1
J	-0.5404950	-0.7765905	0.2702475	2.0586923	1
K	-0.2702475	1.0354540	1.8909363	1.1115528	2
L	0.2702475	-1.0354540	1.1115528	1.8909363	1
M	0.5404950	0.7765905	2.0586923	0.2702475	2
N	1.0809900	-0.5177270	1.9093617	1.3222297	2

(c) Rescale both variables using **min-max normalisation** and perform k-means clustering.

Solution: After the min-max scaling and performing k-mean clustering, if we obtain the final centroids:

$$\boldsymbol{c}_1 = (0.2, 0.2), \quad \boldsymbol{c}_2 = (0.8, 0.8),$$

the final table is:

	x_1'	x_2'	dist1	dist2	cluster
A	0.0	0.0	0.2828427	1.1313708	1
B	0.0	0.2	0.2000000	1.0000000	1
C	0.2	0.0	0.2000000	1.0000000	1
D	0.2	0.2	0.0000000	0.8485281	1
E	0.8	0.8	0.8485281	0.0000000	2
F	0.8	1.0	1.0000000	0.2000000	2
G	1.0	0.8	1.0000000	0.2000000	2
H	1.0	1.0	1.1313708	0.2828427	2
I	0.1	0.7	0.5099020	0.7071068	1
J	0.3	0.2	0.1000000	0.7810250	1
K	0.4	0.9	0.7280110	0.4123106	2
L	0.6	0.1	0.4123106	0.7280110	1
M	0.7	0.8	0.7810250	0.1000000	2
N	0.9	0.3	0.7071068	0.5099020	2

Example 9.1.5 (Final Exam May 2019, Q1(a))**.** (i) Normalisation is normally applied to the algorithm involved distance measures such as clustering. State the reason why normalization is required and how it helps. (2 marks)

Solution: The distance measures are affected by the scale of the variables. By putting all variables into the same range (normalize), the variables will be weighted equally.

(ii) State two types of normalization. Explain on how the normalisation works. (4 marks)

Solution: Min-max normalisation transforms all variables into an interval [0,1] by using the equation (1.2).

Standardisation transforms all variables to a standard scale with mean of zero and standard deviation of one by using the equation (1.1).

Example 9.1.6 (Final Exam Jan 2019, Q1(b))**.** A factory doing oranges packaging would like to group their oranges according to oranges' weight and diameter. Table Q1(b) - i shows the weight and diameter of the oranges.

Orange	Weight (g)	Diameter (cm)
1	108	7.26
2	81	4.61
3	132	6.92
4	118	5.02
5	126	4.16
6	94	5.12
7	150	5.53
8	86	5.10

Table Q1(b) - i

With standardisation, k-means clustering has been performed to group the oranges into three clusters (A, B, C). Table Q1(b) - ii shows part of the results in final iteration of this k-means clustering.

Orange	Distance A	Distance B	Distance C
1	2.3802	2.3082	0.5228
2	2.1092	0.3959	2.8035
3	1.8604	2.6118	0.5228
4	0.5645	1.2903	1.9111
5	0.7206	1.7746	2.7141
6	1.5644	0.3334	2.1141
7	0.9674	2.6736	1.9039
8	1.8927	0.1504	2.3164

Table Q1(b) - ii

(i) Group the oranges into appropriate cluster and state the centroids for each cluster formed. (6 marks)

Solution: Group into cluster with shortest distance and then standardise the predictors, weight, w and diameter, d:

$\mu_w = 111.8750$; $\sigma_w = 24.0620$; $\mu_w = 5.4650$; $\sigma_d = 1.0841$

Orange	Dist_A	Dist_B	Dist_C	Cluster	w^*	d^*
1	2.3802	2.3082	0.5228	C	-0.1610	1.6557
2	2.1092	0.3959	2.8035	B	-1.2831	-0.7886
3	1.8604	2.6118	0.5228	C	0.8364	1.3421
4	0.5645	1.2903	1.9111	A	0.2546	-0.4105
5	0.7206	1.7746	2.7141	A	0.5870	-1.2037
6	1.5644	0.3334	2.1141	B	-0.7429	-0.3182
7	0.9674	2.6736	1.9039	A	1.5844	0.0600
8	1.8927	0.1504	2.3164	B	-1.0753	-0.3367

Compute centroids for each cluster:

$$C_A = (0.8087,\ -0.5181), \quad C_B = (-1.0338,\ -0.4812) \quad C_C = (0.3377,\ 1.4989)$$

(ii) Based on (i), calculate the within cluster sum of square (wss) for each cluster. (6 marks)

Solution: Using the within cluster sum of square (wss) for kth cluster (9.1), we have
Cluster A, $WSS_A = 0.5645^2 + 0.7206^2 + 0.9674^2 = 1.7738$
Cluster B, $WSS_B = 0.3959^2 + 0.3334^2 + 0.1504^2 = 0.2905$
Cluster C, $WSS_C = 0.5228^2 + 0.5228^2 = 0.5466$

According to `https://stats.stackexchange.com/questions/112698/using-k-means-with-other-met` k-means with other distance is not sound. However, it is still possible to just perform calculations using Lloyd algorithm like Example 9.1.2 and the following example despite the final centroids are **not theoretical guarantee to be optimum**.

Example 9.1.7 (Tutorial 2, Q3)**.** The table below shows the marks for 2 assessments (Test and Assignment) of 6 students in a class.

Student	Test	Assignment	Std_Test	Std_Asgmt
1	89	34	0.8119	-0.2871
2	45	27	-1.5435	-1.2145
3	56	30	-0.9546	-0.8170
4	89	44	0.8119	1.0379
5	81	46	0.3836	1.3028
6	83	36	0.4907	-0.0221

Group the students into 3 clusters based on the mark for these 6 students using **Manhattan distance** and the following initial centroids:

$$C_1 = (-0.3658,\ -0.7508); \quad C_2 = (-0.0714,\ 0.1104); \quad C_3 = (0.4372,\ 0.6404).$$

Solution: By using the initial centroids, calculate the distance of all points to the centroids by using Manhattan distance (2.5) to obtain:

Step 1:

Student	Dist.1	Dist.1	Dist.1	Cluster
1	1.6414	1.2807	1.3022	2
2	1.6414	2.7970	3.8355	1
3	0.6551	1.8107	2.8492	1
4	2.9663	1.8107	0.7722	3
5	2.8031	1.6474	0.7160	3
6	1.5852	0.6946	0.7160	2

Update centroids:

$$C_1' = (-1.2491,\ -1.0158); \quad C_2' = (0.6513,\ -0.1546); \quad C_3' = (0.5978,\ 1.1703)$$

Step 2:

Student	Dist.1	Dist.2	Dist.3	Cluster
1	2.7897	0.2931	1.6715	2
2	0.4932	3.2547	4.5261	1
3	0.4932	2.2684	3.5398	1
4	4.1146	1.3530	0.3466	3
5	3.9513	1.7251	0.3466	3
6	2.7335	0.2931	1.2995	2

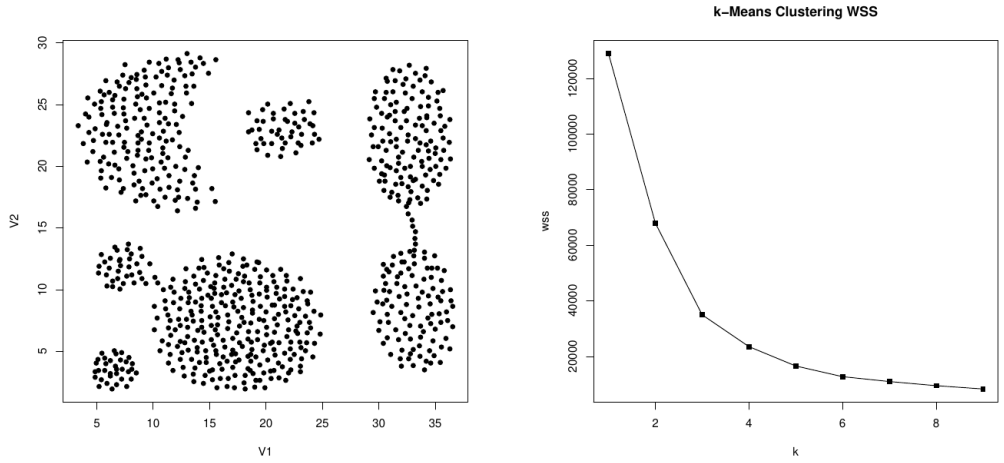
Since the clusters are the same, the stable/final centroids are

$$C_1'' = (-1.2491,\ -1.0158); \quad C_2'' = (0.6513,\ -0.1546); \quad C_3'' = (0.5978,\ 1.1703)$$

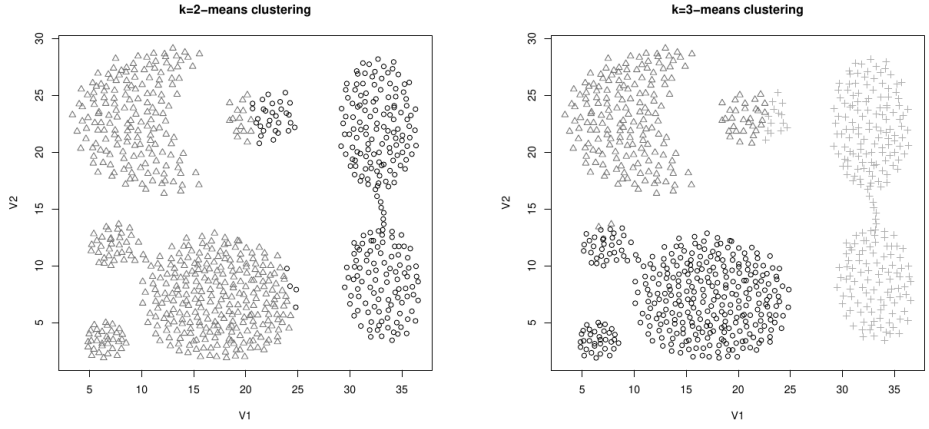
One of the requirements to perform k-means clustering is a pre-determined number of clusters. In some cases, we can have a known number of clusters but this does not apply to all. When we don't know the number of clusters, we want a technique to determine the number of clusters.

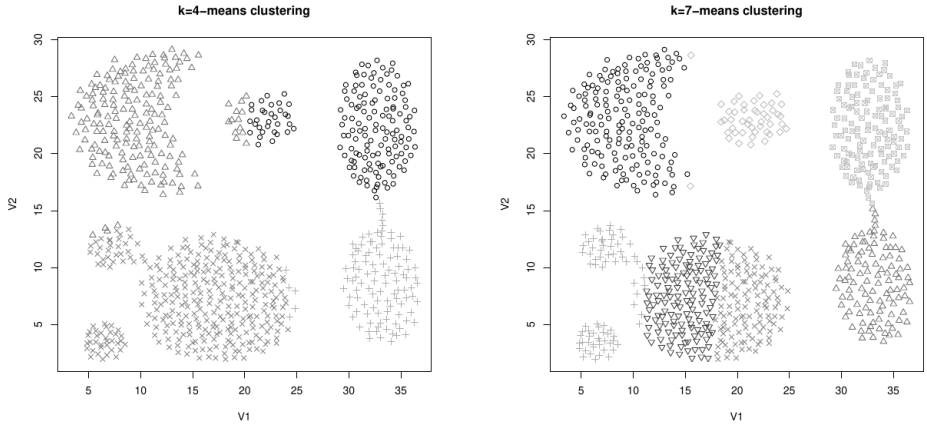
Since one of the properties of good clustering is high intra-cluster (within-cluster) similarity — Observations within clusters will be close together when plotted geometrically. We can make use of this property to determine the optimum number of clusters to be formed. An optimum number of clusters is the one which the *total within groups sum of squares (wss)* (9.1) decreases dramatically as compared to the previous number, and only decreased a little as compared to the after number. This is called the ***elbow principle*** (supported by Python's `https://www.scikit-yb.org/`).

Example 9.1.8 (Elbow principle is not science)**.** Consider the following data and the variation of the total within sum of squares on k clusters.



The elbow principle seems to suggest $k = 3$ or $k = 4$ is the right choice but the actual k-means below seem to suggest that either $k = 3$ or $k = 4$ may not be satisfactory.





The clusters obtain from k-means clustering are not very satisfactory.

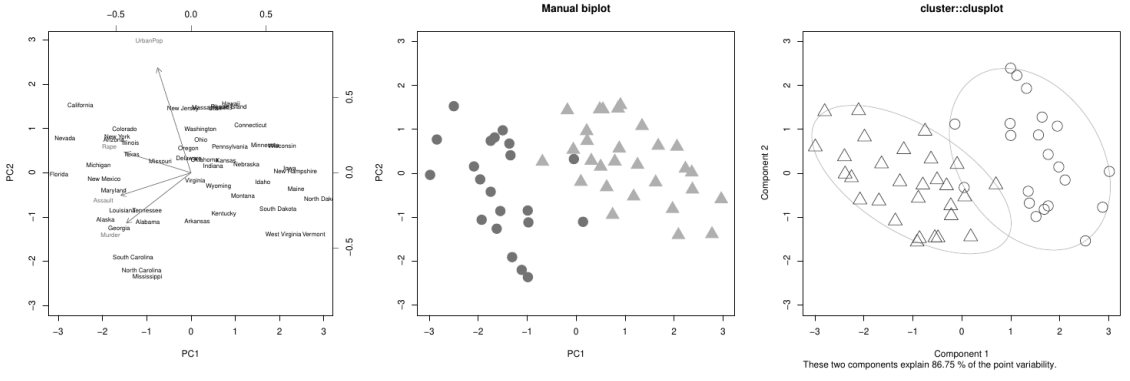
An alternative to the elbow principle is the *silhouette method* (supported by Python in `sklearn.metrics`).

To visualise the clustering result of high-dimensional clusters, we usually project them to 2-D using PCA's biplot.

Example 9.1.9 (4D data)**.** Investigate the clusters in USArrest data using k-means with $k = 2$ and project the clusters into the first and second principle components.

We compare two ways of projecting k-means clusters in R as follows.

```
X = USArrests
pca = prcomp(X, scale=TRUE)    # Build-in standardisation
#
# Standard biplot does not support k-means cluster labels
#
biplot(pca, cex=0.7, scale=0)  # biplot projects data to PC1 & PC2
#
# Manual biplot using scatter-plot
#
kmeans.m = kmeans(X, centers=2)  # centers=k
pdf("usarrests_kmeans1.pdf")
Cl = kmeans.m$cluster
plot(pca$x[,1], pca$x[,2], pch=Cl+15, col=Cl+1, main="Manual biplot",
  cex=2.5, xlim=c(-3,3), ylim=c(-3,3), xlab="PC1", ylab="PC2")
library(cluster)
pdf("usarrests_kmeans2.pdf")
clusplot(X, Cl, main="cluster::clusplot", cex=2.5,
  xlim=c(-3,3), ylim=c(-3,3))
```



9.1.2 k-Prototype Clustering

The k-means clustering only deals with numeric data. The *k-Modes clustering* deals with categorical data while the *k-Prototype clustering* merges k-means and k-modes.

The applications involving mixed data is the 'spatial clustering' in hotspot detection:

- pollution analysis: Malaysia's major concerned, a million of Selangor residents were facing the cease of water supply due to water pollution in Sept 2020 during COVID-19 pendanmic;
- disease analysis: Flu, influensa, Denggi, foot-and-mouth disease;
- crime analysis;
- fire analysis, etc.

In R, k-prototype clustering is implemented in clustMixType. However, the implementation is slower than snail for any data having more than a hundred rows.

```
clustMixType::kproto(x, k, lambda=NULL, iter.max=100, nstart=1,
  na.rm=TRUE, keep.data=TRUE, verbose=TRUE, ...)
```

9.1.3 k-Medoids: PAM, CLARA, CLARANS

The **k-medoids algorithm** is a clustering approach related to k-means clustering for partitioning a data set into k clusters. In k-medoids clustering, each cluster is represented by one of the data point called **cluster medoid** rather than the centroid. Because k-medoids minimises a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, it is more robust (less sensitive) to noise and outliers than k-means.

Partitioning Around Medoids (PAM) algorithm is the most popular k-medoids algorithm:

1. Select k objects to become the medoids, or in case these objects were provided use them as the medoids;
2. Calculate the dissimilarity matrix if it was not provided;
3. Assign every object to its closest medoid;
4. For each cluster search if any of the object of the cluster decreases the average dissimilarity coefficient; if it does, select the entity that decreases this coefficient the most as the medoid for this cluster;
5. If at least one medoid has changed go to 3., else end the algorithm.

Despite being more robust, PAM is only efficient for small data sets but does not scale well for large data sets due to the $O(k(n-k)^2)$ complexity for each iteration.

```
cluster::pam(x, k, diss = inherits(x, "dist"),
  metric = c("euclidean", "manhattan"), medoids = NULL, stand = FALSE,
  cluster.only = FALSE, do.swap = TRUE,
  keep.diss = !diss && !cluster.only && n < 100,
  keep.data = !diss && !cluster.only, pamonce = FALSE, trace.lev = 0)
```

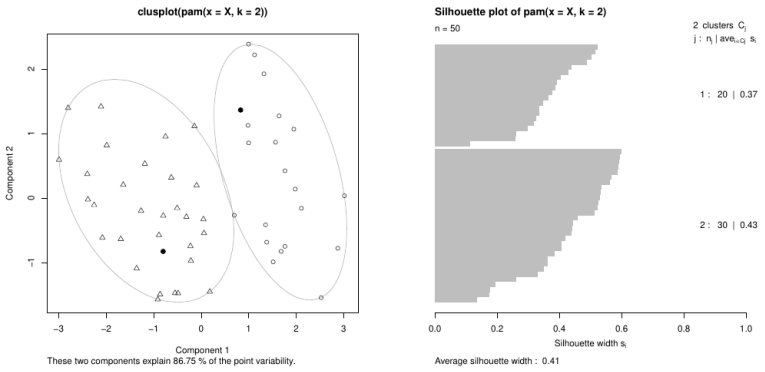
Example 9.1.10 (4D data)**.** Investigate the clusters in USArrest data using PAM with $k = 2$ by projecting the clusters into the first two principle components and draw the silhouette plot.

Solution: The R script

```
# https://www.datanovia.com/en/lessons/k-medoids-in-r-algorithm-and-practical-exam
X = scale(USArrests)
library(cluster)
```

```
pam.m = pam(X, k=2)
pdf("usarrests_pam.pdf")
plot(pam.m, ask=FALSE, which.plots=1)   # Same as clusplot(pam.m)
pdf("usarrests_pam_silh.pdf")
plot(pam.m, ask=FALSE, which.plots=2)   # call: plot.partition
```

generates the PAM($k = 2$) and the silhouette plot shows that $k = 2$ is a good choice.



CLARA and CLARANS are approximating k-medoid methods which tackle PAM's inability to handle large data sets. CLARA applies PAM on multiple subsamples, keeping the best result. CLARANS works on the entire data set, but only explores a subset of the possible swaps of medoids and non-medoids using sampling. BanditPAM uses the concept of multi-armed bandits to choose candidate swaps instead of uniform sampling as in CLARANS.

CLARA = Clustering LARge Applications (Kaufmann and Rousseeuw in 1990):

- Draw multiple samples of the data set, apply PAM on each sample, give the best clustering
- Perform better than PAM in larger data sets
- Efficiency depends on the sample size — A good clustering on samples may not be a good clustering of the whole data set

```
cluster::clara(x, k, metric = c("euclidean", "manhattan", "jaccard"),
  stand = FALSE, cluster.only = FALSE, samples = 5, trace = 0,
  sampsize = min(n, 40 + 2 * k), medoids.x = TRUE, pamLike = FALSE,
  keep.data = medoids.x, rngR = FALSE, correct.d = TRUE)
```

CLARANS = Clustering Large Applications based upon RAndomized Search

- The problem space graph of clustering
  - A vertex is k from n numbers, $\binom{n}{k}$ vertices in total
  - PAM searches the whole graph
  - CLARA searches some random sub-graphs
- CLARANS climbs hills
  - Randomly sample a set and select k medoids
  - Consider neighbours of medoids as candidate for new medoids
  - Use the sample set to verify
  - Repeat multiple times to avoid bad samples

9.1.4 Fuzzy clustering: FANNY, FLAME

Fuzzy clustering (also known as *soft clustering* or *soft k-means*) is a form of clustering in which each data point can belong to more than one cluster. It is different from k-means and k-medoid clustering (known as hard or non-fuzzy clustering), where each object is affected exactly to one cluster.

In fuzzy clustering, points close to the centre of a cluster, may be in the cluster to a higher degree than points in the edge of a cluster. The degree, to which an element belongs to a given cluster, is a numerical value varying from 0 to 1.

One of the most widely used fuzzy clustering algorithms is the FANNY (Fuzzy ANalysis) C-means clustering (FCM) algorithm.

- Select an initial fuzzy pseudo-partition, i.e., assign values to all the w_{ij}
- Repeat
  - Compute the centroid of each cluster using the fuzzy pseudo-partition
  - Recompute the fuzzy pseudo-partition, i.e., the w_{ij}
- Until the centroids do not change (or the change is below some threshold)

Critical Details of the Algorithm:

- Optimisation on sum of the squared error (SSE):

$$SSE(C_1, ..., C_k) = \sum_{j=1}^{k} \sum_{i=1}^{n} w_{ij}^p d(\boldsymbol{x}_i, \boldsymbol{c}_j)^2$$

- Computing centroids: $\boldsymbol{c}_j = \sum_{i=1}^{n} w_{ij}^p \boldsymbol{x}_i / \sum_{i=1}^{n} w_{ij}^p$
- Updating the fuzzy pseudo-partition

$$w_{ij} = \left(\frac{1}{d(\boldsymbol{x}_i, \boldsymbol{c}_j)^2} \right)^{1/(p-1)} \Big/ \sum_{q=1}^{k} \left(\frac{1}{d(\boldsymbol{x}_i, \boldsymbol{c}_q)^2} \right)^{1/(p-1)}$$

The characteristic of p in the Algorithm:

- When $p \to 1$, FCM behaves like traditional k-means;
- When p is larger, the cluster centroids approach the global centroid of all data points;
- The partition becomes fuzzier as p increases.

```
# R
cluster::fanny(x, k, diss = inherits(x, "dist"), memb.exp = 2,
  metric = c("euclidean", "manhattan", "SqEuclidean"),
  stand = FALSE, iniMem.p = NULL, cluster.only = FALSE,
  keep.diss = !diss && !cluster.only && n < 100,
  keep.data = !diss && !cluster.only,
  maxit = 500, tol = 1e-15, trace.lev = 0)
```

Another fuzzy clustering algorithm is the *Fuzzy clustering by Local Approximation of MEmberships (FLAME)*. It defines clusters in the dense parts of a dataset and performs cluster assignment solely based on the neighbourhood relationships among objects. The key feature of this algorithm is that the neighbourhood relationships among neighbouring objects in the feature space are used to constrain the memberships of neighbouring objects in the fuzzy membership space.

9.1.5 Mixture Model: GMM

A **mixture model** is a mixture of k component distributions that collectively make a mixture distribution:

$$f(x) = \sum_{i=1}^{k} \alpha_i f_i(x). \tag{9.2}$$

The α_i represents a mixing weight for the ith component where $\sum_{i=1}^{k} \alpha_i = 1$. The $f_i(x)$ components in principle are arbitrary in the sense that you can choose any sort of distribution.

For continuous case, it has the form

$$g(x) = \int_{\Theta} f(x|\alpha)\rho(\alpha)d\alpha.$$

In practice, parametric distribution (e.g. gaussians), are often used since a lot work has been done to understand their behaviour.

If we substitute each $f_i(x)$ for a gaussian we get the **gaussian mixture models (GMM)**:

$$f(\boldsymbol{X} = \boldsymbol{x}, Y = k) = \sum_{i=1}^{k} \alpha_i \frac{1}{\sqrt{(2\pi)^p |\Sigma_i|}} \exp(-\frac{1}{2}(\boldsymbol{x} - \mu_i)\Sigma_i^{-1}(\boldsymbol{x} - \mu_i)^T).$$

Likewise, if we substitute each $f_k(x)$ for a binomial distribution, we get a *binomial mixture model (BMM)*.

GMM is available in R's mclust [Scrucca et al., 2016] and mixtools (plotGMM can be used for plotting) libraries and in Python's sklearn.mixture submodule.

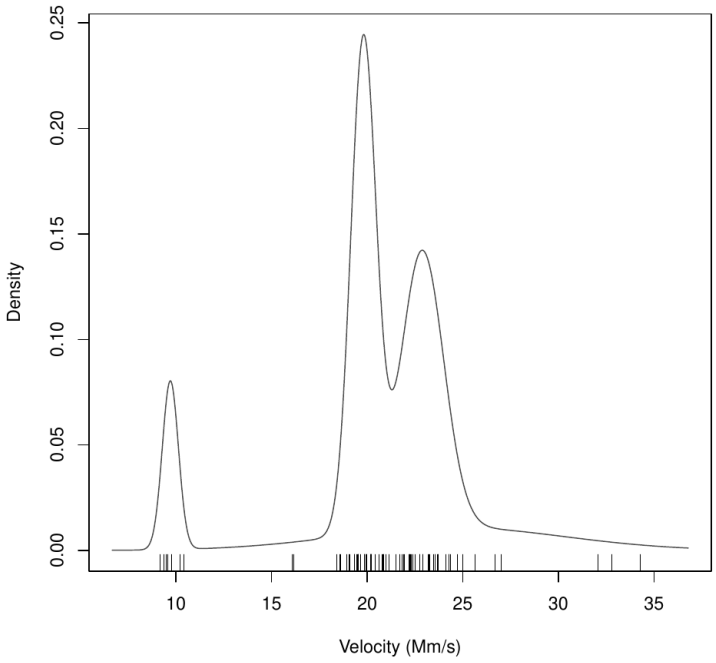
```
# R
mclust::Mclust(data, G = NULL, modelNames = NULL, prior = NULL,
  control = emControl(), initialization = NULL,
  warn = mclust.options("warn"), x = NULL,
  verbose = interactive(), ...)
# G: An integer vector specifying the numbers of mixture components
# (clusters) for which the BIC is to be calculated.  The default is 1:9

# Python
sklearn.mixture.GaussianMixture(n_components=1, *,
  covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100,
  n_init=1, init_params='kmeans', weights_init=None, means_init=None,
  precisions_init=None, random_state=None, warm_start=False,
  verbose=0, verbose_interval=10)
```

Example 9.1.11 (1D galaxy signal)**.** Use GMM to analyse the galaxies dataset from MASS.

Solution: Since the data is 1D, we write a script to compare the histogram to the probability densities found by the GMM:

```
1  # https://cran.r-project.org/web/packages/sBIC/vignettes/GaussianMixtures.pdf
2  library(MASS)
3  X = galaxies/1000
4  library(mclust, quietly=TRUE)
5  pdf("gmm_eg.pdf",width=12,height=8)
6  par(mfrow=c(1,2))
7  hist(X, prob=TRUE, ylim=c(0,0.15)); lines(density(X))
8  gmm.model = Mclust(X, G=4, model="V")
9  summary(gmm.model)
10 plot(gmm.model, what="density", main="", xlab="Velocity (Mm/s)")
11 rug(X)
```

The result is as follows.

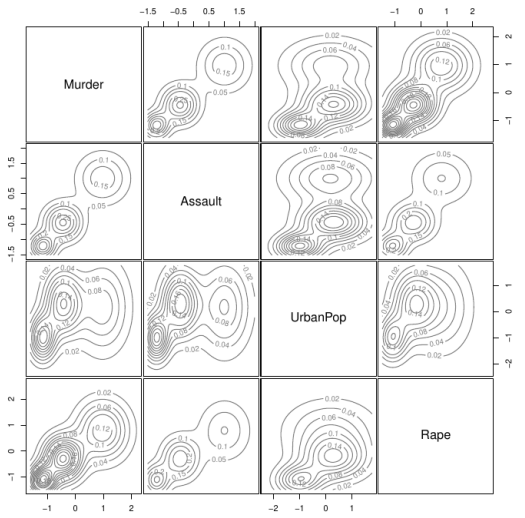


Example 9.1.12 (4D data USArrest with GMM)**.** Use standardisation and GMM to analyse the distributions in the USArrest data.

Solution: The R script

```
X = scale(USArrests)
library(mclust, quietly=TRUE)
gmm.model = Mclust(X)
print(summary(gmm.model))
plot(gmm.model, what="density")
```

generates the following analysis report and the projection of the distribution of 4-D data for all pairs of variables.




```
----------------------------------------
Gaussian finite mixture model fitted by
EM algorithm
----------------------------------------

Mclust VEI (diagonal, equal shape) model
with 3 components:

 log-likelihood  n df       BIC
ICL
      -217.3636 50 20 -512.9677 -517.5878

Clustering table:
 1  2  3
20 10 20
```

The **EM algorithm** stands for the Expectation Maximisation which tries to fit the data with the mixture model (9.2) by finding the maximum likelihood function. In `mclust::Mclust`, the `emControl` provides controls on the EM algorithm to GMM.

An outline of the EM algorithm:

1. Select an initial set of model parameters
2. Repeat
   (a) Expectation Step (E-step): for each object, calculate the probability that it belongs to each distribution i, i.e., $\mathbb{P}(x_i|i)$
   (b) Maximisation Step (M-step): given the probabilities from the expectation step, find the new estimates of the parameters that maximise the expected likelihood
3. Until the parameters are stable

A general implementation is given in the R's `EMCluster` package.
For the Gaussian mixture model, we assume a generative process for the data as follows:

$$\boldsymbol{x}_i|c_i \sim Normal(\mu_{c_i}, \Sigma_{c_i}), \quad c_i \sim Discrete(p)$$

where c_i are one of the $1, ..., K$ class, p is some probability vector. The EM algorithm seeks to maximise the likelihood

$$f(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n|p, \mu, \Sigma) = \prod_{i=1}^{n} f(\boldsymbol{x}_i|p, \mu, \Sigma)$$

over all parameters p, $\mu_1, ..., \mu_K, \Sigma_1, ..., \Sigma_K$ using the cluster assignments $c_1, ..., c_n$ as the hidden data. The step 2 which maximises the log-likelihood becomes

(2a) E-step: For $i = 1, ..., n$, set

$$\phi_i(k) = \frac{p_k N(x_i|\mu_k, \Sigma_k)}{\sum_j p_j N(x_i|\mu_j, \Sigma_j)}, \quad k = 1, \ldots, K.$$

(2b) M-step: For $k = 1, \ldots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and update the values

$$p_k = \frac{n_k}{n}, \quad \mu_k = \sum_{i=1}^{n} \phi_i(k)\boldsymbol{x}_i, \quad \Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T.$$

Mixture models (MM) are more general than k-means and FANNY clustering and the clusters can be characterised by a small number of parameters. The results may satisfy the statistical assumptions of the generative models.

However, MM need large data sets to identify the right probability densities. The worst problems of MM are that they are computationally expensive and it is hard to estimate the number of clusters.

9.2 Affinity Propagation

In statistics and data mining, **affinity propagation (AP)** is a clustering algorithm based on the concept of "message passing" between data points. Unlike clustering algorithms such as k-means or k-medoids, AP does not require the number of clusters to be determined or estimated before running the algorithm. Similar to k-medoids, affinity propagation finds "exemplars", members of the input set that are representative of clusters.

```
# R
apcluster::apcluster(s, x, p=NA, q=NA, maxits=1000, convits=100,
  lam=0.9, includeSim=FALSE, details=FALSE, nonoise=FALSE, seed=NA)

# Python
sklearn.cluster.AffinityPropagation(*, damping=0.5, max_iter=200,
  convergence_iter=15, copy=True, preference=None,
  affinity='euclidean', verbose=False, random_state=None)
```

The inventors of AP showed it is better for certain computer vision and computational biology tasks, e.g. clustering of pictures of human faces and identifying regulated transcripts, than k-means. A study comparing AP and Markov clustering on protein interaction graph partitioning found Markov clustering to work better for that problem.

9.3 Mean Shift

Mean shift is a non-parametric feature-space mathematical analysis technique for locating the maxima of a density function. The application domains include cluster analysis in computer vision and image processing. It is an iterative method where we start with an initial estimate $\boldsymbol{x}$. Let a kernel function $K(\boldsymbol{x}_i - \boldsymbol{x})$ be given. This function determines the weight of nearby points for re-estimation of the mean. Typically a Gaussian kernel on the distance to the current estimate is used, $K(\boldsymbol{x}_i - \boldsymbol{x}) = e^{-c\|\boldsymbol{x}_i - \boldsymbol{x}\|}$. The weighted mean of the density in the window determined by K is

$$m(\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}_i \in N(\boldsymbol{x})} K(\boldsymbol{x}_i - \boldsymbol{x})\boldsymbol{x}_i}{\sum_{\boldsymbol{x}_i \in N(\boldsymbol{x})} K(\boldsymbol{x}_i - \boldsymbol{x})}$$

where $N(\boldsymbol{x})$ is the neighbourhood of $\boldsymbol{x}$, a set of points for which $K(\boldsymbol{x}_i - \boldsymbol{x}) \neq 0$.

The difference $m(\boldsymbol{x}) - \boldsymbol{x}$ is called mean shift. The mean-shift algorithm now sets

$$\boldsymbol{x} \leftarrow m(\boldsymbol{x}),$$

and repeats the estimation until $m(\boldsymbol{x})$ converges.

```
# R
LPCM::ms(X, h, subset,  thr=0.01, scaled= 1, iter=200, plot=TRUE, ...)

# Python
sklearn.cluster.MeanShift(*, bandwidth=None, seeds=None,
  bin_seeding=False, min_bin_freq=1, cluster_all=True,
  n_jobs=None, max_iter=300)
```

For clustering, consider a set of points in two-dimensional space. Assume a circular window centred at C and having radius r as the kernel. Mean-shift is a hill climbing algorithm which involves shifting this kernel iteratively to a higher density region until convergence. Every shift is defined by a mean shift vector. The mean shift vector always points toward the direction of the maximum increase in the density. At every iteration the kernel is shifted to the centroid or the mean of the points within it. The method of calculating this mean depends on the choice of the kernel. In this case if a Gaussian kernel is chosen instead of a flat kernel, then every point will first be assigned a weight which will decay exponentially as the distance from the kernel's centre increases. At convergence, there will be no direction at which a shift can accommodate more points inside the kernel.

For visual tracking, we would create a confidence map in the new image based on the colour histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position. The confidence map is a probability density function on

the new image, assigning each pixel of the new image a probability, which is the probability of the pixel colour occurring in the object in the previous image. A few algorithms, such as kernel-based object tracking, ensemble tracking, CAMshift expand on this idea.

9.4 Spectral Clustering

Spectral clustering make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. The similarity matrix is provided as an input and consists of a quantitative assessment of the relative similarity of each pair of points in the dataset. In application to image segmentation, spectral clustering is known as segmentation-based object categorisation.

The spectral clustering algorithm can be broken down into 4 basic steps.

1. Construct a similarity/adjacency matrix $A = [a_{ij}]$ for the similarity graph of the data X where $a_{ij} \geq 0$ is the similarity between data points with indices i and j;
2. Determine the degree matrix $D = [d_{ii}]$, $d_{ii} = \sum_j a_{ij}$ and the Laplacian matrix $L := D - A$;
3. Compute the eigenvectors of the matrix L;
4. Using the second smallest eigenvector as input, train a k-means model and use it to classify the data.

An implementation of the algorithm in R can be found in `http://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html` listed below.

```
1  # http://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html
2  library(mlbench)
3  set.seed(111)
4  obj <- mlbench.spirals(100,1,0.025)
5  my.data <-  4 * obj$x
6  #plot(my.data)
7  # Step 1. Construct a similarity graph
8  make.similarity <- function(my.data, similarity) {
9    N <- nrow(my.data)
10   S <- matrix(rep(NA,N^2), ncol=N)
11   for(i in 1:N) {
12     for(j in 1:N) {
13       S[i,j] <- similarity(my.data[i,], my.data[j,])
14     }
15   }
16   S
17 }
18 S <- make.similarity(my.data, function(x1, x2, alpha=1) {
19   exp(- alpha * norm(as.matrix(x1-x2), type="F"))
20 })
21 # Step 2. Determine the adjacency/affinity matrix A, degree matrix D and
22 # the Laplacian matrix L
23 make.affinity <- function(S, n.neighboors=2) {
24   N <- length(S[,1])
25
26   if (n.neighboors >= N) {  # fully connected
27     A <- S
28   } else {
29     A <- matrix(rep(0,N^2), ncol=N)
30     for(i in 1:N) { # for each line
31       # only connect to those points with larger similarity
```

```
      best.similarities <- sort(S[i,], decreasing=TRUE)[1:n.neighboors]
      for (s in best.similarities) {
        j <- which(S[i,] == s)
        A[i,j] <- S[i,j] # to make an undirected graph, i.e.
        A[j,i] <- S[i,j] # the matrix becomes symmetric
      }
    }
  }
  A
}
A <- make.affinity(S, 3)  # use 3 neighboors (includes self)
D <- diag(apply(A, 1, sum)) # sum rows
U <- D - A
# matrix power operator: computes M^power (M must be diagonalizable)
"%^%" <- function(M, power)
  with(eigen(M), vectors %*% (values^power * solve(vectors)))
k   <- 2
evL <- eigen(U, symmetric=TRUE)
Z   <- evL$vectors[,(ncol(evL$vectors)-k+1):ncol(evL$vectors)]
# Step 4. Using the second smallest eigenvector as input, train
# a k-means model and use it to classify the data.
km = kmeans(Z, centers=k, nstart=5)
plot(my.data, col=km$cluster, pch=km$cluster)
```

The above algorithms demonstrates the inner workings of the spectral clustering algorithm. In real-world application, we use standard implementations listed below.

```
# R
kernlab::specc(x, data = NULL, na.action = na.omit, ...)

kknn::specClust(data, centers=NULL, nn = 7, method = "symmetric",
  gmax=NULL, ...)

# Python
sklearn.cluster.SpectralClustering(n_clusters=8, *, eigen_solver=None,
  n_components=None, random_state=None, n_init=10, gamma=1.0,
  affinity='rbf', n_neighbors=10, eigen_tol=0.0, assign_labels='kmeans',
  degree=3, coef0=1, kernel_params=None, n_jobs=None, verbose=False)
```

The following example illustrates the difference between k-means algorithm and spectral clustering algorithm.

Example 9.4.1. Given the spiral data X generated below.

```
library(mlbench)
set.seed(111)
X = 4*mlbench.spirals(100,1,0.025)$x
colnames(X) = c('x','y')
```

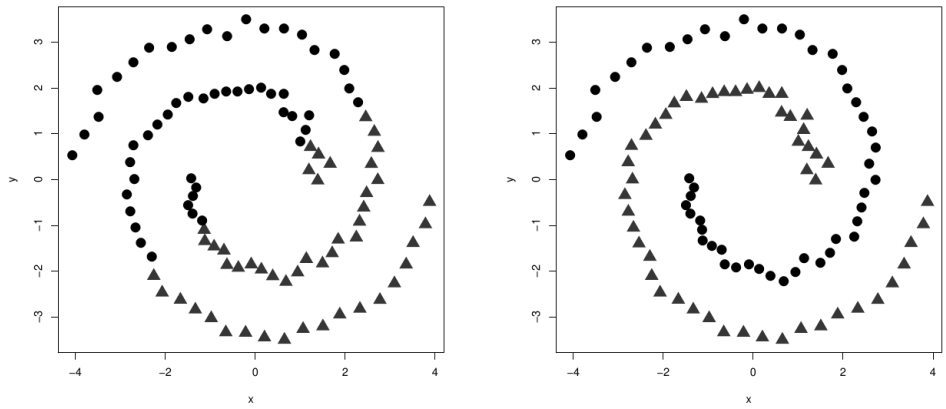
Compare the clusters between k-means and spectral clustering.

Solution: By writing an R script

```
1 library(mlbench)
2 set.seed(111)
3 X = 4*mlbench.spirals(100,1,0.025)$x
4 colnames(X) = c('x','y')
5 km = kmeans(X, 2)
6 pdf("clust_kmean.pdf")
```

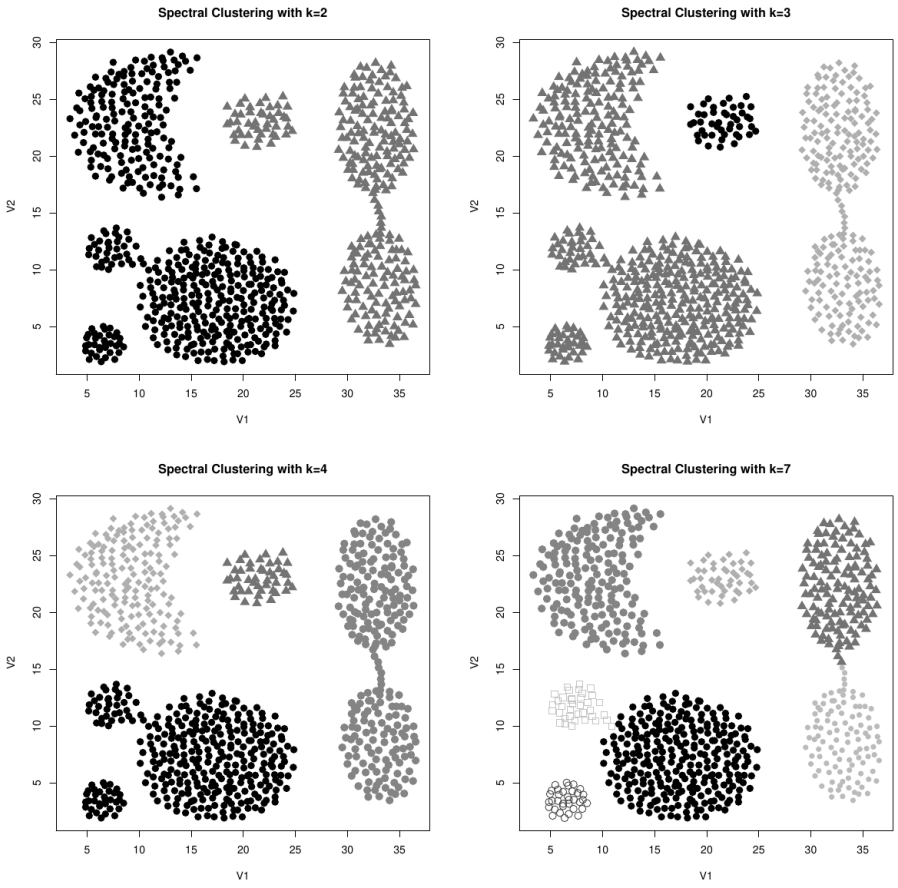
```
plot(X, col=km$cluster, pch=15+km$cluster, cex=2)

library(kernlab)
sc = specc(X, centers=2)
pdf("clust_spectral.pdf")
plot(X, col=sc, pch=15+sc, cex=2)
```

we obtain an expected clustering using spectral clustering but getting a not very satisfactory clustering using k-means.



Example 9.4.2. Analyse the data in Example 9.1.8 using spectral clustering.

Solution: By following Example 9.1.8 using $k = 2$, 2, 4 and 7 we obtain a much more satisfactory clusters compare to the k-means.



9.5 Density-Based Methods

The **drawbacks** of distance-based methods (such as the partitioning methods and the hierarchical methods) are: (a) hard to find clusters with irregular shapes; (b) hard to specify the number of clusters; (c) a cluster must be dense but distance-based methods may find clusters which are not dense.

The rationale of **density-based clustering** is that a cluster is composed of well-connected dense region, while objects in sparse areas are removed as noises. The advantage of density-based clustering is that it can filter out noise and find clusters of arbitrary shapes (as long as they are composed of connected dense regions). It can find irregular clusters by dividing the whole space into many small areas such that the density of an area can be estimated and dense area is likely in a cluster. Start from a dense area, traverse connected dense areas and discover clusters in irregular shape.

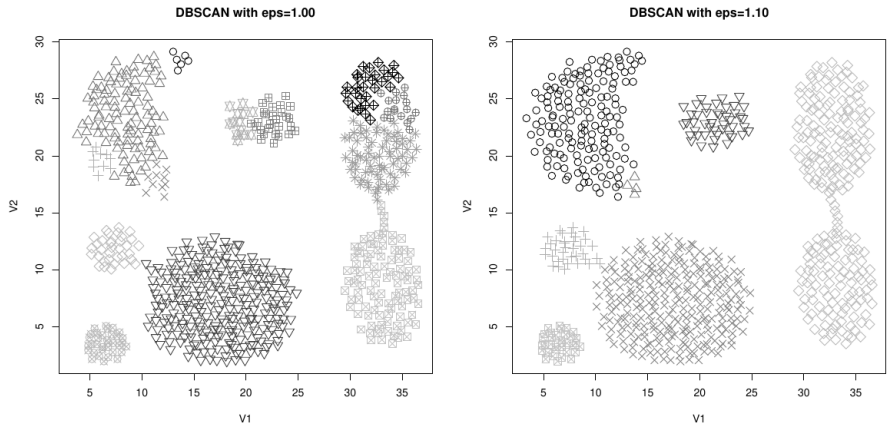
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) clustering is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbours), marking as outliers points that lie alone in low-density regions (whose nearest neighbours are too far away).

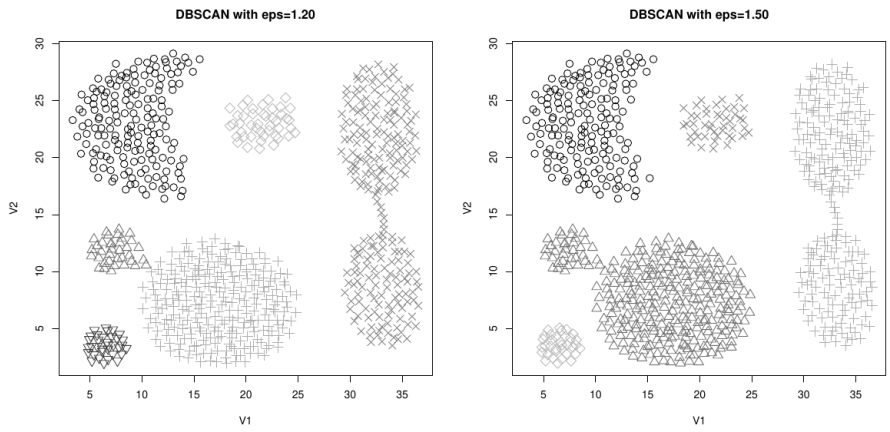
```
# R
dbscan::dbscan(x, eps, minPts=5, weights=NULL, borderPoints=TRUE, ...)
```

Example 9.5.1. Analyse the clusters for the data in Example 9.1.8 using DBSCAN.

Solution: In contrast to partition methods in which need to specify the number of clusters. For DBSCAN, we need to specify the 'density'. Being too large make all points into one cluster while being too small creates too many clusters. A sample R script analysing the data is given below.

```
X = read.table("Aggregation.txt", sep="\t", header=F)
X$V3 = NULL
X = as.matrix(X)
density = c(1,1.1,1.2,1.5)
for(k in 1:length(density)){
    d = density[k]
    sc = dbscan::dbscan(X, eps=d); print(sc)
    pdf(sprintf("aggr_dbscan%d.pdf", k))
    plot(X, col=sc$cluster, pch=sc$cluster, cex=1.5,
         main=sprintf("DBSCAN with eps=%.2f",d))
}
```





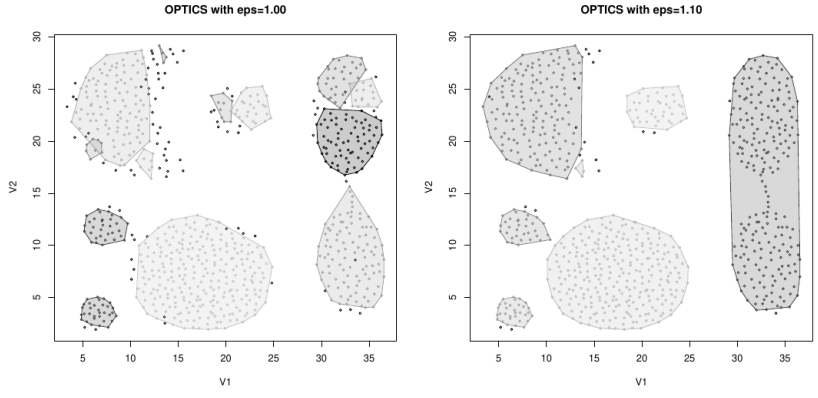
OPTICS (Ordering Points To Identify the Clustering Structure) is an algorithm for finding density-based clusters in spatial data. It was presented by Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel and Jörg Sander. Its basic idea is similar to DBSCAN, but it addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density. To do so, the points of the database are (linearly) ordered such that spatially closest points become neighbours in the ordering. Additionally, a special distance is stored for each point that represents the density that must be accepted for a cluster so that both points belong to the same cluster.

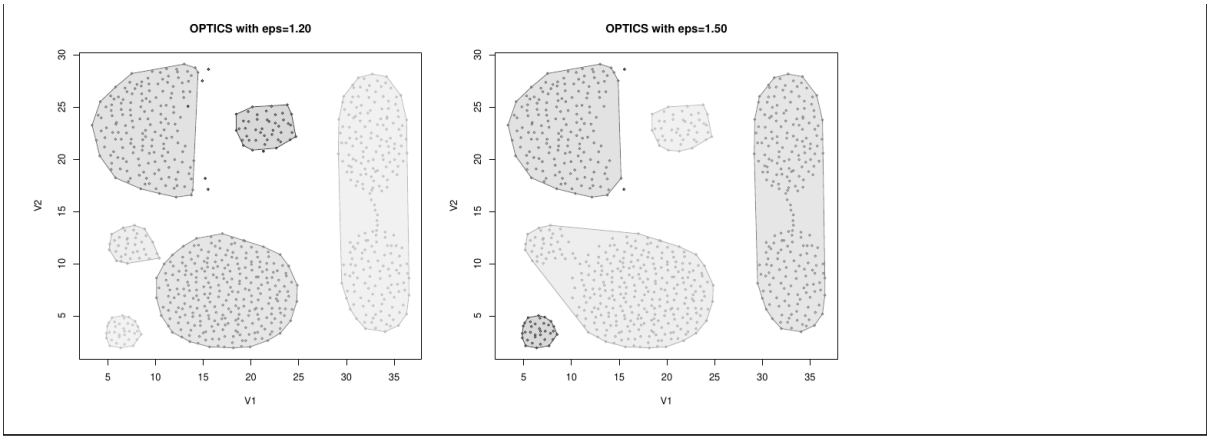
```
# R
dbscan::optics(x, eps = NULL, minPts = 5, ...)  # difficult to use
```

Example 9.5.2. Analyse the clusters for the data in Example 9.1.8 using OPTICS.

Solution: OPTICS is more complex than DBSCAN.

```
X = read.table("Aggregation.txt", sep="\t", header=F)
X$V3 = NULL
X = as.matrix(X)
library(dbscan)
density = c(1,1.1,1.2,1.5)
for(k in 1:length(density)){
    d = density[k]
    op = optics(X, eps=d)
    pdf(sprintf("aggr_optics%d.pdf", k))
    res = extractDBSCAN(op, eps_cl = d)
    hullplot(X, res, main=sprintf("OPTICS with eps=%.2f",d))
}
```





9.6 Grid-Based Methods

Grid-based clustering methods quantise the object space into a finite number of cells (hyper-rectangles) and then perform the required operations on the quantised space. The main advantage of Grid based method is its fast processing time which depends on number of cells in each dimension in quantised space. A few grid based methods from academic research are CLIQUE (CLustering In QUEst), STING (STatistical INformation Grid), MAFIA (Merging of Adaptive Intervals Approach to Spatial Data Mining), Wave Cluster and O-CLUSTER (Orthogonal partitioning CLUSTERing).

CLIQUE partitions the data space and finds the number of points that lie inside each cell of the partition.

STING divides the spatial area into rectangular cells. [Wang et al., 1997]

Wave Cluster uses performs multi-resolution clustering using the wavelet method [Sheikholeslami et al., 1988].

9.7 Hierarchical Clustering

Hierarchical clustering merges or splits data successively to form a tree known as ***dendrogram*** [Hegland, 2004]. Two broad classes of hierarchical clustering algorithms are [Hegland, 2004]

- *agglomerative*: start with single points and join them together whenever they are close. E.g ; Agglomerative Nesting (AGNES) with various linkages, BIRCH, CURE, ROCK, Chameleon;
- *divisive*: move from the top down and break the clusters up when they are dissimilar. E.g. Divisia Analysis (DIANA), MOVA.

Hierarchical clustering, spectral clustering, etc. relies on the distance matrix which will be stated.

9.7.1 Distance Matrix / Proximity Matrix

Suppose we have n data samples $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})$, $i = 1, \cdots, n$. Stacking the sample data $\boldsymbol{x}_i$ leads to a matrix

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & \ddots & & x_{2p} \\ \vdots & & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

where x_{ij} is the ith sample/observation for the jth variable/feature.

The **distance matrix** or **proximity matrix** between observations of data $\boldsymbol{X}$ is used in the computation of AGNES and it can be obtained for a dissimilarity function d as

$$D = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix} \tag{9.3}$$

where $d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $h, k = 1, \cdots, n$.

Distance Matrix in R

The basic distance matrix in R has the form:

```
dist(x, method = "euclidean", diag = FALSE, upper = FALSE, p = 2)
```

The options for `method` are 'euclidean', 'maximum', 'manhattan', 'canberra, 'binary', 'minkowski'.

The `daisy` function supports euclidean, manhattan and gower.

```
cluster::daisy(x, metric = c("euclidean","manhattan","gower"),
  stand=FALSE, type=list(), weights=rep.int(1,p), warnBin = warnType,
  warnAsym = warnType, warnConst = warnType, warnType = TRUE)
```

When 'stand' is set TRUE and all 'x's are numeric, the measurements in 'x' are standardised before calculating the dissimilarities.

Distance Matrix in Python

The Scikit-learn library has a richer variety of distance measures.

```
sklearn.metrics.pairwise_distances(X, Y=None, metric='euclidean', *,
  n_jobs=None, force_all_finite=True, **kwds)
```

The "metric" options include `euclidean`/`l2`, `manhattan`/`l1`/`cityblock`, `cosine`, `braycurtis`, `canberra`, `chebyshev`, `correlation`, `dice`, `hamming`, `jaccard`, `kulsinski`, `mahalanobis`, `minkowski`, `rogestanimoto`, `russellrao`, `seuclidean`, `sokalmichener`, `sokalsneath`, `yule`, `sqeuclidean`.

Example 9.7.1. You are given the following observations with two variables, x_1 and x_2 in Example 9.1.3.

Obs	x_1	x_2
A	1	1
B	1	2
C	2	1
D	2	2
E	8	8
F	8	9
G	9	8
H	9	9

Calculate the Euclidean distance between each pair of observations and construct a distance matrix.

Solution: Note that $p = 2$ and $n = 8$ (8 observations).

$$d_{hh} = \sqrt{(x_{h1} - x_{h1})^2 + (x_{h2} - x_{h2})^2} = 0, \quad h = 1, \cdots, n;$$

$$\begin{aligned}
d_{21} &= \sqrt{(x_{21} - x_{11})^2 + (x_{22} - x_{12})^2} = \sqrt{(1-1)^2 + (2-1)^2} = 1 \\
d_{31} &= \sqrt{(x_{31} - x_{11})^2 + (x_{32} - x_{12})^2} = \sqrt{(2-1)^2 + (1-1)^2} = 1 \\
d_{41} &= \sqrt{(x_{41} - x_{11})^2 + (x_{42} - x_{12})^2} = \sqrt{(2-1)^2 + (2-1)^2} = \sqrt{2} \\
d_{51} &= \sqrt{(x_{51} - x_{11})^2 + (x_{52} - x_{12})^2} = \sqrt{(8-1)^2 + (8-1)^2} = \sqrt{98} \\
&\vdots
\end{aligned}$$

After all distances have been calculated, we have the distance matrix

```
df = data.frame(x1=c(1,1,2,2,8,8,9,9),x2=c(1,2,1,2,8,9,8,9))
dist(df)
          1         2         3         4         5         6         7
2  1.000000
3  1.000000  1.414214
4  1.414214  1.000000  1.000000
5  9.899495  9.219544  9.219544  8.485281
6 10.630146  9.899495 10.000000  9.219544  1.000000
7 10.630146 10.000000  9.899495  9.219544  1.000000  1.414214
8 11.313708 10.630146 10.630146  9.899495  1.414214  1.000000  1.000000
```

Example 9.7.2. Calculate the Manhattan distance between each pair of observations from Example 9.7.1 and construct a distance matrix.

For hierarchical clustering to work, we need the extension for "dissimilarity" between pairs of clusters. This leads to the notion of **linkage** — a measure for dissimilarity between cluster A and cluster B (let $\boldsymbol{c}_X$ be the centroid of cluster X):

Linkage	Update formula for $d(I \cup J, K)$	Cluster Dissimiliarty
Minimum or single-linkage	$\min\{d(I,J), d(J,K)\}$	$\min\{\, d(a,b) \;:\; a \in A,\, b \in B \,\}$
Maximum or complete-linkage	$\max\{d(I,J), d(J,K)\}$	$\max\{\, d(a,b) : a \in A,\, b \in B \,\}$
(Unweighted) average linkage (or UPGMA)	$\dfrac{n_I d(I,K) + n_J d(J,K)}{n_I + n_J}$	$\dfrac{1}{\lvert A\rvert \cdot \lvert B\rvert} \displaystyle\sum_{a \in A} \sum_{b \in B} d(a,b)$
Weighted average linkage (or WPGMA)	$\dfrac{d(I,K) + d(J,K)}{2}$	-
Centroid or UPGMC	$\sqrt{\dfrac{n_I d(I,K) + n_J d(J,K)}{n_I + n_J} - \dfrac{n_I n_J d(I,J)}{(n_I + n_J)^2}}$	$\lVert \boldsymbol{c}_A - \boldsymbol{c}_B \rVert_2$
Median or WPGMC	$\sqrt{\dfrac{d(I,K)}{2} + \dfrac{d(J,K)}{2} - \dfrac{d(I,J)}{4}}$	-
Ward	$\sqrt{\dfrac{(n_I + n_K) d(I,K) + (n_J + n_K) d(J,K) - n_K d(I,J)}{n_I + n_J + n_K}}$	$\sqrt{\dfrac{2\lvert A\rvert\lvert B\rvert}{\lvert A\rvert + \lvert B\rvert}} \lVert \boldsymbol{c}_A - \boldsymbol{c}_B \rVert_2$
Minimum energy clustering	-	$\frac{2}{nm} \sum_{i,j=1}^{n,m} \lVert a_i - b_j \rVert_2 - \frac{1}{n^2} \sum_{i,j=1}^{n} \lVert a_i - a_j \rVert_2 - \frac{1}{m^2} \sum_{i,j=1}^{m} \lVert b_i - b_j \rVert_2$

Except the minimum energy clustering, all linkages can be more-or-less unified in the Lance-Wiliams combinatorial formula:

$$d(I \cup J, K) = a(I)d(I,K) + a(J)d(J,K) + bd(I,J) + c|d(I,K) - d(J,K)|$$

where a, b and c are dependent on the linkage, I, J are indices of the clusters just merged and K is the index for other points or clusters.

Linkage	$a(I)$	$a(J)$	b	c
Minimum or single-linkage	0.5	0.5	0	-0.5
Maximum or complete-linkage	0.5	0.5	0	0.5
(Unweighted) average linkage (or UPGMA)	$\dfrac{n_I}{n_I + n_J}$	$\dfrac{n_J}{n_I + n_J}$	0	0
Weighted average linkage (or WPGMA)	0.5	0.5	0	0
Centroid or UPGMC	$\dfrac{n_I}{n_I + n_J}$	$\dfrac{n_J}{n_I + n_J}$	$-\dfrac{n_I n_J}{(n_I + n_J)^2}$	0
Median or WPGMC	0.5	0.5	-0.25	0
Ward	$\dfrac{n_I + n_K}{n_I + n_J + n_K}$	$\dfrac{n_J + n_K}{n_I + n_J + n_K}$	$\dfrac{n_I + n_J}{n_I + n_J + n_K}$	0

The Ward linkage minimises the sum of squared differences within all clusters and is similar to the k-means objective fuction. The complete linkage minimises distance btween observations of pairs of clusters, average linkage minimises the average of the distances between all observations of paris of clusters while single linkage minimises the distance between the closest observations of pairs of clusters.

The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

Algomerative hierarchical clustering is available in R and Python.

```
# R
hclust(d, method = "complete", members = NULL)
# method = "single", "complete", "average" (=UPGMA), "ward.D", "ward.D2",
#          "mcquitty" (=WPGMA), "median" (=WPGMC), "centroid" (=UPGMC)

plot(x, labels = NULL, hang = 0.1, check = TRUE, axes = TRUE,
  frame.plot = FALSE, ann = TRUE, main = "Cluster Dendrogram",
  sub = NULL, xlab = NULL, ylab = "Height", ...)

cluster::agnes(x, diss = inherits(x, "dist"), metric = "euclidean",
  stand = FALSE, method = "average", par.method,
  keep.diss = n < 100, keep.data = !diss, trace.lev = 0)

# Python
sklearn.cluster.AgglomerativeClustering(n_clusters=2, *,
  affinity='euclidean', memory=None, connectivity=None,
  compute_full_tree='auto', linkage='ward', distance_threshold=None,
  compute_distances=False)
# linkage = 'ward', 'complete', 'average', 'single'
```

The `plot` will draw the dendrogram for the classification tree generated by the agglomerative hierarchical clustering algorithm as follows:

1. Begin with n observations and a distance measure of all the pairwise dissimilarities. Treat each observation as its own cluster.

2. Examine all inter-cluster dissimilarities and identify the pair of clusters that are least dissimilar (most similar). Fuse the two clusters. Repeat this step until all clusters are fused.

9.7.2 Complete Linkage, Single Linkage and Average Linkage (UPGMA)

The **complete linkage** defines the dissimilarity between groups of observations as

$$d_{(hk)l} = \max\{d_{hl},\ d_{kl}\}, \quad l \neq h, k, \quad 1 \leq l \leq n. \tag{9.4}$$

The method computes all pairwise dissimilarities between the elements in a cluster and the elements in another cluster, and considers the *largest value* of these dissimilarities, i.e. (9.4), as the distance between the two clusters.

Example 9.7.3 (Wikipedia: Complete-Linkage Clustering)**.** Based on a JC69 genetic distance matrix computed from the 5S ribosomal RNA sequence alignment of five bacteria: Bacillus subtilis (A), Bacillus stearothermophilus (B), Lactobacillus viridescens (C), Acholeplasma modicum (D), and Micrococcus luteus (E).

	A	B	C	D	E
A	0	17	21	31	23
B	17	0	30	34	21
C	21	30	0	28	39
D	31	34	28	0	43
E	23	21	39	43	0

Perform complete-linkage hierarchical clustering.

Solution: Step 1: The (nonzero) smallest distance of the table is $d(A, B) = 17$. Therefore, we cluster A and B leading to the table below using the complete-linkage (9.4):

	A,B	C	D	E
A,B	0	**30**	**34**	**23**
C	**30**	0	28	39
D	**34**	28	0	43
E	**23**	39	43	0

A and B's branch length is $d(A, B) = 17$

Step 2: The smallest distance of the above table is $d((A, B), E) = 23$ and we cluster (A,B) and E leading to the table below using the complete-linkage:

	A,B,E	C	D
A,B,E	0	**39**	**43**
C	**39**	0	28
D	**43**	28	0

(A,B) and E's branch length is $d((A, B), E) - d(A, B) = 23 - 17 = 6$.

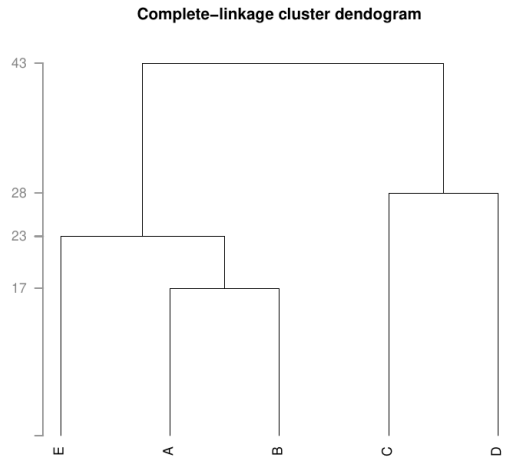
Step 3: The smallest distance of the above table is $d(C, D) = 28$ and we cluster C and D leading to the table below using the complete-linkage:

	A,B,E	C,D
A,B,E	0	**43**
C,D	**43**	0

Root r to (A,B,E)'s branch length is $d(r, (A, B, E)) = d((A, B, E), (C, D)) - d((A, B), E) = 43 - 23 = 20$.

Root r to (C,D)'s branch length is $d(r, (C, D)) = d((A, B, E), (C, D)) - d(C, D) = 43 - 28 = 15$.

These data allows us to construct the dendrogram below (compare to the Wikipedia):



Example 9.7.4 (Final Exam May 2019, Q1(b))**.** Table Q1(b) – i shows the data collected from five observations with three variables, x_1, x_2 and x_3.

Observation	x_1	x_2	x_3
A	6	3	4
B	3	2	1
C	7	5	5
D	3	1	2
E	2	3	2

Table Q1(b) – i

Table Q1(b) – ii shows parts of the Euclidean distance matrix computed from data collected in Table Q1(b) – i after applying standardisation.

Observation	A	B	C	D	E
A		2.3880		2.2835	
B				0.9082	
C	1.5496	3.6635		3.7430	
D					1.4251
E	2.2104		3.2358		

Table Q1(b) – ii

(i) Based on the information given, complete the distance matrix shown in Table Q1(b) – ii. (4 marks)

Solution: First, we compute the mean and sample standard deviation of x_i using `sapply(df,mean)`, `sapply(df,sd)`:

$$\overline{X} = (4.2,\ 2.8,\ 2.8); \quad s_X = (2.1679,\ 1.4832,\ 1.6432).$$

Then apply standardisation to observations B and E using `scale(df)`:

Observation	x_1	x_2	x_3
B	-0.5535	-0.5394	-1.0954
E	-1.0148	0.1348	-0.4869

Finally, we compute the Euclidean distance between B and E and complete the **distance matrix** using `dist(scale(df))`:

$$d_{BE} = \sqrt{(x_{1B} - x_{1E})^2 + (x_{2B} - x_{2E})^2 + (x_{3B} - x_{3E})^2} = 1.0187$$

	A	B	C	D	E
A	0				
B	2.3880	0			
C	1.5496	3.6635	0		
D	2.2835	0.9082	3.7430	0	
E	2.2104	1.0187	3.2358	1.4251	0

(ii) Group the observations using hierarchical clustering with **complete linkage**. Sketch the dendrogram formed by the hierarchical clustering. Group the observations into two groups and state the observations inside each group. (6 marks)

Solution: Step 1: $d(B, D) = 0.9082$

	(B, D)	A	C	E
(B, D)	0			
A	2.3880	0		
C	3.7430	1.5496	0	
E	1.4251	2.2104	3.2358	0

Step 2: $d(BD, E) = 1.4251$

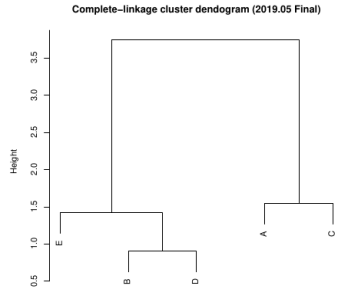
	(BD, E)	A	C
(BD, E)	0		
A	2.3880	0	
C	3.7430	1.5496	0

Step 3: $d(A, C) = 1.5496$

	(A, C)	(BD, E)
(A, C)	0	
(BD, E)	3.7430	0

Step 4: $d(AC, BDE) = 3.7430$

The dendrogram illustrates the data are "clustered" at two groups:
Group 1 = (B, D, E);
Group 2 = (A, C)



Example 9.7.5 (May 2022 Semester Final Exam, Q3(b))**.** Construct the dendrogram for the hierarchical clustering with **complete linkage** for the data in Table 3.1.

	x_1	x_2	x_3
A	2.6	3	4
B	1.4	4	5
C	2.5	2	2
D	1.7	2	5
E	2.7	3	4
F	2.4	4	4

Table 3.1: Three-dimensional data for clustering.

Suppose the Minkowski distance with $r = 1.5$ is used and the incomplete distance table for the points A to F is obtained as follows:

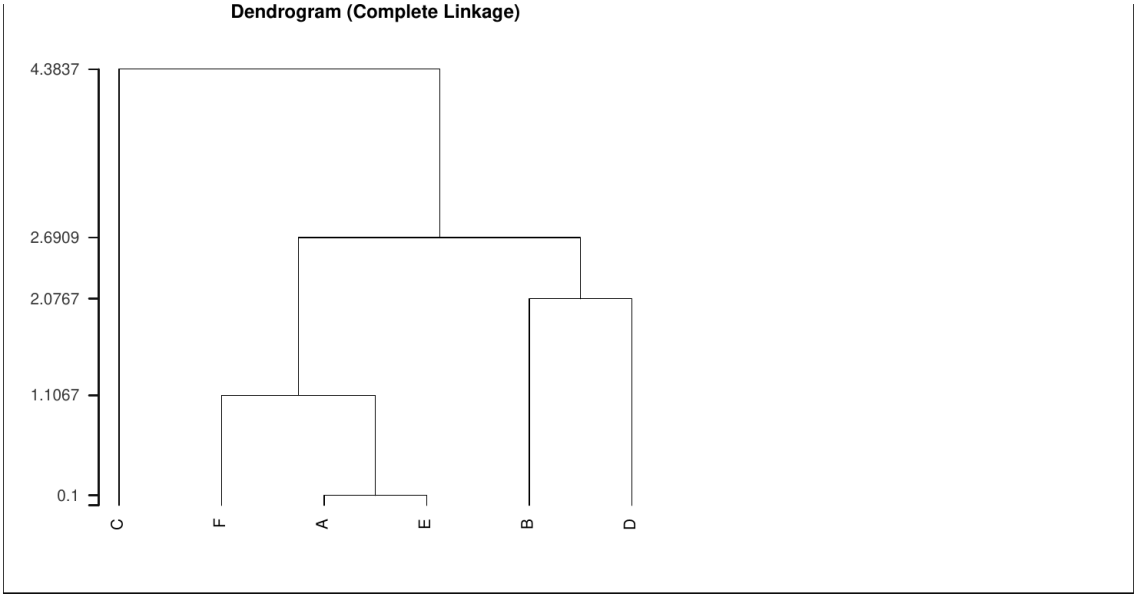
```
      A      B      C      D
B 2.2230
C 2.4607 4.3837
D 2.0120 2.0767 3.2694
E 0.1000 2.2974 2.4852 2.0801
F 1.0588 1.5874 3.1866 2.6909
```

(i) Construct the complete distance matrix for the points A to F using a Minkowski distance with $r = 1.5$ on the data from the Table 3.1 by calculating the distance $d(E, F)$. Show the calculation steps properly. (2 marks)

$$d(E,F) = \left[|2.7-2.4|^{1.5} + |3-4|^{1.5} + |4-4|^{1.5}\right]^{1/1.5} = 1.106745 \qquad [1.5 \text{ marks}]$$

```
      A      B      C      D      E      F
A 0
B 2.2230 0
C 2.4607 4.3837 0
D 2.0120 2.0767 3.2694 0
E 0.1000 2.2974 2.4852 2.0801 0
F 1.0588 1.5874 3.1866 2.6909 1.1067 0              [0.5 mark]
```

(ii) Perform the necessary steps to draw the dendrogram with proper labels for the hierarchical clustering with complete linkage for the data from the Table 3.1. If you are able to find the correct value for $d(E,F)$, use the value from part (i) to do this part. Otherwise, use the Manhattan distance to approximate $d(E,F)$ to work on this part. (9 marks)



Example 9.7.6 (May 2022 Semester Final Exam, Q3(c))**.** When the clustering methods mentioned in Example 9.1.1 and Example 9.7.5 are applied to the study of a real-world data, is there a need to scale the data? Justify your answer. (2 marks)

Yes. There is a need to scale the data to prevent some predictors/features to dominant the data due to the magnification of predictors/features in the distance function. . . [2 marks]

Example 9.7.7 (May 2023 Semester Final Exam, Q4(b))**.**

Given the unlabelled data in Table 4.2.

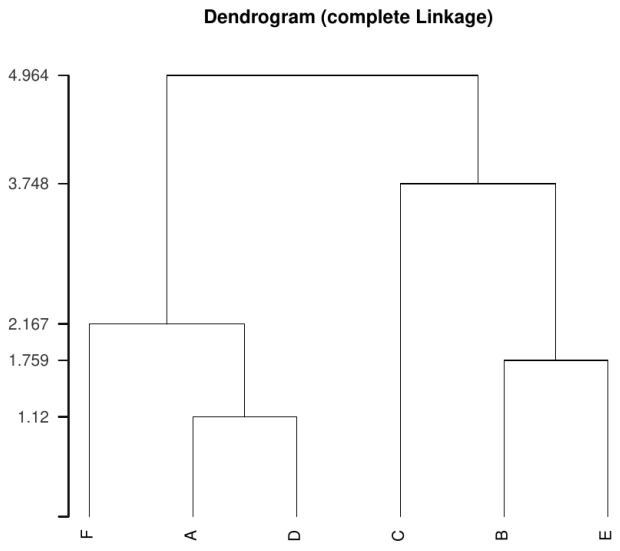
Obs.	x_1	x_2
A	2.37	1.72
B	1.03	6.50
C	−0.31	3.00
D	2.22	2.83
E	1.55	4.82
F	4.14	2.97

Table 4.2: Unlabelled data.

Suppose the (Euclidean) distance matrix of Table 4.2 is given below:

```
        A       B       C       D       E       F
A   0
B   4.964   0
C   2.970   3.748   0
D   1.120   3.858   2.536   0
E   3.207   1.759   2.602   2.100   0
F   2.167   4.705   4.450   1.925   3.183       0
```

Construct a hierarchical clustering with **Euclidean distance** and **complete linkage** and then draw the **dendrogram** of the hierarchical clustering. (9 marks)



The **single linkage** defines the dissimilarity between the (h, k) groups of observations as follows:

$$d_{(hk)l} = \min\{d_{hl}, d_{kl}\}, \quad l \neq h, k, \quad 1 \leq l \leq n \tag{9.5}$$

The method computes all pairwise dissimilarities between two clusters and considers the *smallest* of these similarities as a linkage criterion.

Example 9.7.8. The scores (1=worst to 10=best) of 8 children in three subjects were recorded.

Child	A	B	C	D	E	F	G	H
Maths	7	4	6	10	6	5	9	10
Music	5	7	8	10	2	9	6	9
History	8	8	6	9	2	4	8	9

Construct a distance matrix among the children. Perform hierarchical clustering process by using single linkage.

Solution: The **distance matrix** among children A to H is

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{bmatrix} 0. & & & & & & & \\ 3.6056 & 0. & & & & & & \\ 3.7417 & 3. & 0. & & & & & \\ 5.9161 & 6.7823 & 5.3852 & 0. & & & & \\ 6.7823 & 8.0623 & 7.2111 & 11.3578 & 0. & & & \\ 6. & 4.5826 & 2.4495 & 7.1414 & 7.3485 & 0. & & \\ 2.2361 & 5.099 & 4.1231 & 4.2426 & 7.8102 & 6.4031 & 0. & \\ 5.099 & 6.4031 & 5.099 & 1. & 10.6771 & 7.0711 & 3.3166 & 0. \end{bmatrix}.$$

Note that $d(H, D) = 1$ is the minimum distance. So H and D are most similar and we "cluster" them and calculate:

- $d(\{H, D\}, A) = \min\{d(H, A), d(D, A)\} = 5.099$
- $d(\{H, D\}, B) = \min\{d(H, B), d(D, B)\} = 6.4031$
- $d(\{H, D\}, C) = \min\{d(H, C), d(D, C)\} = 5.099$
- $d(\{H, D\}, E) = \min\{d(H, E), d(D, E)\} = 10.6771$
- $d(\{H, D\}, F) = \min\{d(H, F), d(D, F)\} = 7.0711$
- $d(\{H, D\}, G) = \min\{d(H, G), d(D, G)\} = 3.3166$

The step 1 distance matrix becomes

$$\begin{array}{c} \\ A \\ B \\ C \\ D, H \\ E \\ F \\ G \end{array} \begin{bmatrix} 0. & & & & & & \\ 3.6056 & 0. & & & & & \\ 3.7417 & 3. & 0. & & & & \\ 5.099 & 6.4031 & 5.099 & 0. & & & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. & & \\ 6. & 4.5826 & 2.4495 & 7.0711 & 7.3485 & 0. & \\ 2.2361 & 5.099 & 4.1231 & 3.1666 & 7.8102 & 6.4031 & 0. \end{bmatrix}.$$

The smallest distance in step 1 distance matrix is $d(A, G) = 2.2361$, therefore A and G will be grouped together and single linkage calculations will produce a step 2 distance matrix:

$$\begin{array}{c} \\ A, G \\ B \\ C \\ D, H \\ E \\ F \end{array} \begin{bmatrix} 0. & & & & & \\ 3.6056 & 0. & & & & \\ 3.7417 & 3. & 0. & & & \\ 3.3166 & 6.4031 & 5.099 & 0. & & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. & \\ 6. & 4.5826 & 2.4495 & 7.0711 & 7.3485 & 0. \end{bmatrix}.$$

The smallest distance in step 2 distance matrix is $d(C, F) = 2.4495$, therefore C and F will be grouped together and single linkage calculations will produce a step 3 distance

matrix:

$$\begin{matrix} A,G \\ B \\ C,F \\ D,H \\ E \end{matrix} \begin{bmatrix} 0. & & & & \\ 3.6056 & 0. & & & \\ 3.7417 & 3. & 0. & & \\ 3.3166 & 6.4031 & 5.099 & 0. & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. \end{bmatrix}.$$

Note that $d(\{A,D,G,F\},B) = \min\{3.6056, 4.5826\}$, $d(\{A,D,G,F\},C) = \min\{3.7417, 2.4495\}$.

The smallest distance in step 3 distance matrix is $d(\{C,F\},B) = 3$, therefore {C,F} and B will be grouped together and single linkage calculations will produce a step 4 distance matrix:

$$\begin{matrix} A,G \\ B,C,F \\ D,H \\ E \end{matrix} \begin{bmatrix} 0. & & & \\ 3.6056 & 0. & & \\ 3.3166 & 5.099 & 0. & \\ 6.7823 & 7.2111 & 10.6771 & 0. \end{bmatrix}.$$

The smallest distance in step 4 distance matrix is $d(\{A,G\},\{D,H\}) = 3.3166$, therefore {A,G} and {D,H} will be grouped together and single linkage calculations will produce a step 5 distance matrix:

$$\begin{matrix} A,G,D,H \\ B,C,F \\ E \end{matrix} \begin{bmatrix} 0. & & \\ 3.6056 & 0. & \\ 6.7823 & 7.2111 & 0. \end{bmatrix}.$$

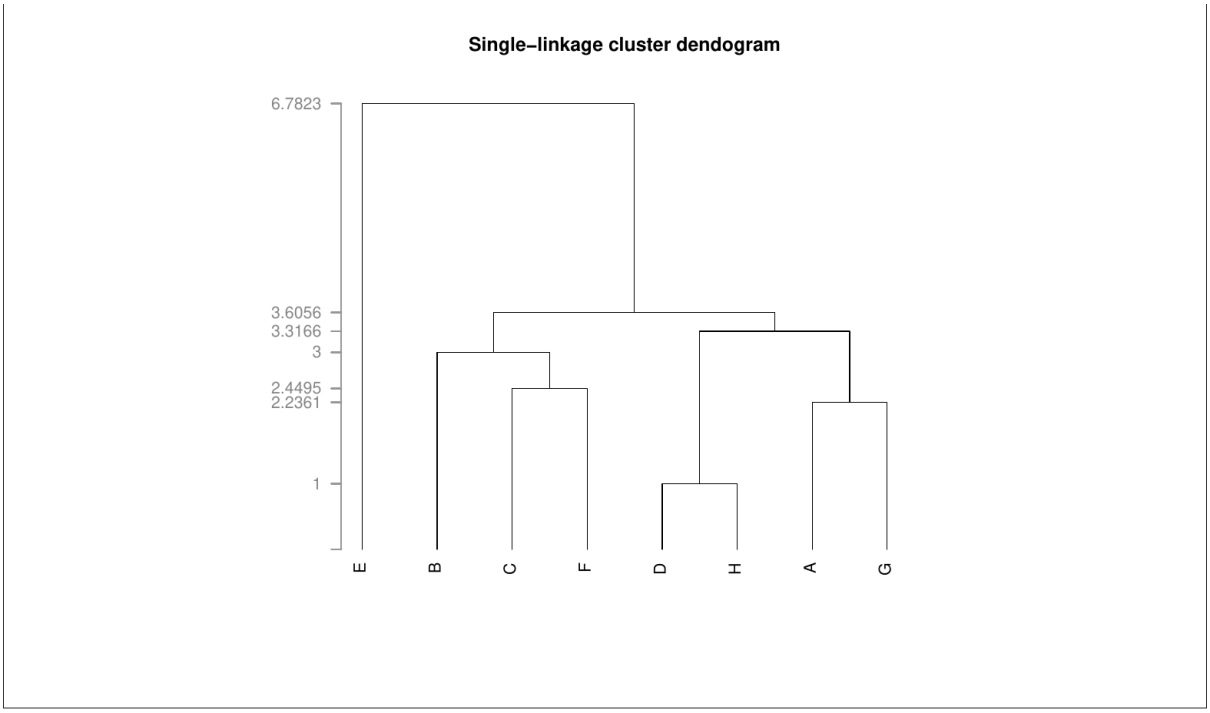
The smallest distance in step 5 distance matrix is $d(\{A,G,D,H\},\{B,C,F\}) = 3.6056$, therefore {A,G,D,H} and {B,C,F} will be grouped together and single linkage calculations will produce a step 6 distance matrix:

$$\begin{matrix} A,G,D,H,B,C,F \\ E \end{matrix} \begin{bmatrix} 0. & \\ 6.7823 & 0. \end{bmatrix}.$$

Using an R script:

```
d.f = data.frame(
    Maths=c(7 ,4 ,6 ,10 ,6 ,5 ,9 ,10),
    Music=c(5 ,7 ,8 ,10 ,2 ,9 ,6 ,9 ),
    Hist=c(8 ,8 ,6 ,9 ,2 ,4 ,8 ,9),
    row.names=c('A','B','C','D','E','F','G','H')
)
#hc = hclust(dist(d.f),method='single')
#plot(hc,xlab="",sub="",hang=-1, axes=FALSE, ylab="",main="")
#title(main="Single-linkage cluster dendogram")
hc = hclust(dist(d.f),method='average')
plot(hc,xlab="",sub="",hang=-1, axes=FALSE, ylab="",main="")
title("Average-linkage cluster dendogram")
axis(side=2, at=c(0,hc$height), col="#F38630", labels=FALSE, lwd=2)
mtext(round(hc$height,4), side=2, at=hc$height, line=1, col="#A38630", las=2)
```

the dendrogram can be plotted nicely as:



The **average linkage** or *Unweighted Pair Group Method with Arithmetic Mean (UPGMA)* defines distance between groups as the average of the distances between all pairs of individuals in the two groups:

$$d_{(hk)l} = \frac{1}{N_{(hk)}N_l} \sum_{i \in S_{(hk)}} \sum_{j \in S_l} d_{ij}$$

where $N_{(hk)}$ and N_l are the number of items in clusters $S_{(hk)}$ and S_l, respectively.

Example 9.7.9. Refer to Example 9.7.8, perform hierarchical clustering process by using (a)

(a) Complete linkage;

Solution: The minimum distance is identified to be $\boxed{d(D, H) = 1}$.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{bmatrix} 0. & & & & & & & \\ 3.6056 & 0. & & & & & & \\ 3.7417 & 3. & 0. & & & & & \\ 5.9161 & 6.7823 & 5.3852 & 0. & & & & \\ 6.7823 & 8.0623 & 7.2111 & 11.3578 & 0. & & & \\ 6. & 4.5826 & 2.4495 & 7.1414 & 7.3485 & 0. & & \\ 2.2361 & 5.099 & 4.1231 & 4.2426 & 7.8102 & 6.4031 & 0. & \\ 5.099 & 6.4031 & 5.099 & 1. & 10.6771 & 7.0711 & 3.3166 & 0. \end{bmatrix}.$$

Step 1: Merge D,H & minimum distance is $\boxed{d(A, G) = 2.2361}$

$$\begin{array}{c} \\ A \\ B \\ C \\ D, H \\ E \\ F \\ G \end{array} \begin{bmatrix} 0. & & & & & & \\ 3.6056 & 0. & & & & & \\ 3.7417 & 3. & 0. & & & & \\ 5.9161 & 6.7823 & 5.3852 & 0. & & & \\ 6.7823 & 8.0623 & 7.2111 & 11.3578 & 0. & & \\ 6. & 4.5826 & 2.4495 & 7.1414 & 7.3485 & 0. & \\ 2.2361 & 5.099 & 4.1231 & 4.2426 & 7.8102 & 6.4031 & 0. \end{bmatrix}.$$

Step 2: Merge A,G & minimum distance is $\boxed{d(C,F) = 2.4495}$

$$\begin{matrix} A,G \\ B \\ C \\ D,H \\ E \\ F \end{matrix} \begin{bmatrix} 0. & & & & & \\ 5.099 & 0. & & & & \\ 4.1231 & 3. & 0. & & & \\ 5.9161 & 6.7823 & 5.3852 & 0. & & \\ 7.8102 & 8.0623 & 7.2111 & 11.3578 & 0. & \\ 6.4031 & 4.5826 & 2.4495 & 7.1414 & 7.3485 & 0. \end{bmatrix}.$$

Step 3: Merge C,F & minimum distance is $\boxed{d(B,(C,F)) = 4.5826}$

$$\begin{matrix} A,G \\ B \\ C,F \\ D,H \\ E \end{matrix} \begin{bmatrix} 0. & & & & \\ 5.099 & 0. & & & \\ 6.4031 & 4.5826 & 0. & & \\ 5.9161 & 6.7823 & 7.1414 & 0. & \\ 7.8102 & 8.0623 & 7.3485 & 11.3578 & 0. \end{bmatrix}.$$

Step 4: Merge B,(C,F) & minimum distance is $\boxed{d(AG,DH) = 5.9161}$

$$\begin{matrix} A,G \\ B,(C,F) \\ D,H \\ E \end{matrix} \begin{bmatrix} 0. & & & \\ 6.4031 & 0. & & \\ 5.9161 & 7.1414 & 0. & \\ 7.8102 & 8.0623 & 11.3578 & 0. \end{bmatrix}.$$

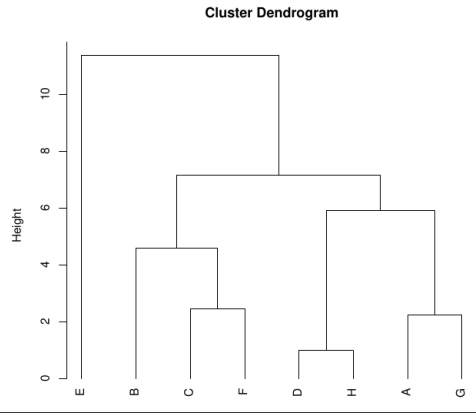
Step 5: Merge (B,(C,F)), (D,H) & minimum distance is $\boxed{d(AG,DH) = 7.1414}$

$$\begin{matrix} AG,DH \\ BCF \\ E \end{matrix} \begin{bmatrix} 0. & & \\ 7.1414 & 0. & \\ 11.3578 & 8.0623 & 0. \end{bmatrix}.$$

Step 6: Merge (A,G), (D,H) & minimum distance is $\boxed{d(AGDHBCF,H) = 11.3578}$

$$\begin{matrix} AGDH,BCF \\ E \end{matrix} \begin{bmatrix} 0. & \\ 11.3578 & 0. \end{bmatrix}.$$

$$\begin{bmatrix} 3. & 7. & 1. & 2. \\ 0. & 6. & 2.2361 & 2. \\ 2. & 5. & 2.4495 & 2. \\ 1. & 10. & 4.5826 & 3. \\ 8. & 9. & 5.9161 & 4. \\ 11. & 12. & 7.1414 & 7. \\ 4. & 13. & 11.3578 & 8. \end{bmatrix}$$



(b) Average linkage (`https://en.wikipedia.org/wiki/UPGMA`)

Solution: From part (a), the minimum distance is $d(D,H)=1$.

Step 1: Merge D,H; calculate new distances and find the min. dist. to be $d(A,G)=2.2361$:

$$d(DH,A)=\frac{1}{2\times 1}(5.9161+5.099)\approx 5.5076,$$
$$d(DH,B)\approx 6.5927,\quad d(DH,C)=5.2421$$
$$d(DH,E)=\frac{1}{2\times 1}(11.3578+10.6771)\approx 11.0175,$$
$$d(DH,F)\approx 7.1063,\quad d(DH,G)\approx 3.7796.$$

$$\begin{array}{c} A\\ B\\ C\\ D,H\\ E\\ F\\ G \end{array}\begin{bmatrix} 0. & & & & & & \\ 3.6056 & 0. & & & & & \\ 3.7417 & 3. & 0. & & & & \\ \boxed{5.5076} & \boxed{6.5927} & \boxed{5.2421} & 0. & & & \\ 6.7823 & 8.0623 & 7.2111 & \boxed{11.0175} & 0. & & \\ 6. & 4.5826 & 2.4495 & \boxed{7.1063} & 7.3485 & 0. & \\ 2.2361 & 5.099 & 4.1231 & \boxed{3.7796} & 7.8102 & 6.4031 & 0. \end{bmatrix}.$$

Step 2: Merge A,G; calculate new distances and find the min. dist. is $d(C,F)=2.4495$

$$d(AG,B)=\frac{1}{2\times 1}(3.6056+5.099)\approx 4.3523,\ d(AG,C)\approx 6.5927,\ d(AG,E)\approx 7.1063,$$

$$d(AG,F)\approx 3.7796,\quad d(AG,DH)=\frac{1}{2\times 2}(d_{AD}+d_{AH}+d_{GD}+d_{GH})=\frac{d(DH,A)+d(DH,G)}{2}$$
$$=\frac{5.5076+3.7796}{2}\approx 4.6436$$

$$\begin{array}{c} A,G\\ B\\ C\\ D,H\\ E\\ F \end{array}\begin{bmatrix} 0. & & & & & \\ \boxed{4.3523} & 0. & & & & \\ \boxed{3.9324} & 3. & 0. & & & \\ \boxed{4.6436} & 6.5927 & 5.2421 & 0. & & \\ \boxed{7.2963} & 8.0623 & 7.2111 & 11.0175 & 0. & \\ \boxed{6.2016} & 4.5826 & 2.4495 & 7.1063 & 7.3485 & 0. \end{bmatrix}.$$

Step 3: Merge C,F; calculate new distances and find the min. dist. is $d(B,CF)=3.7913$:

$$d(CF,AG)=\frac{1}{2\times 2}(d_{CA}+d_{CG}+d_{FA}+d_{FG})=\frac{d(AG,C)+d(AG,F)}{2}\approx 5.0670$$

$$d(CF,DH)=\frac{1}{2\times 2}(d_{CD}+d_{CH}+d_{FD}+d_{FH})=\frac{d(DH,C)+d(DH,F)}{2}\approx 6.1742$$
$$d(CF,B)\approx 3.7913,\quad d(CF,E)\approx 7.2798$$

$$\begin{array}{c} A,G\\ B\\ C,F\\ D,H\\ E \end{array}\begin{bmatrix} 0. & & & & \\ 4.3523 & 0. & & & \\ \boxed{5.0670} & \boxed{3.7913} & 0. & & \\ 4.6436 & 6.5927 & \boxed{6.1742} & 0. & \\ 7.2963 & 8.0623 & \boxed{7.2798} & 11.0175 & 0. \end{bmatrix}.$$

Step 4: Merge B,CF; calculate new distances and find the min. dist. to be $d(AG, DH) = 4.6436$:

$$d(BCF, AG) = \frac{1}{3\times 2}(4d(CF, AG) + 2d(B, AG)) = \frac{2\times 5.0670 + 1\times 4.3523}{3} \approx 4.8288$$

$$d(BCF, DH) = \frac{1}{3\times 2}(4d(CF, DH) + 2d(B, DH)) = \frac{2\times 6.1742 + 6.5927}{3} \approx 6.3137$$

$$d(BCF, E) = (d_{BE}+d_{CE}+d_{FE}) = \frac{1}{3\times 1}(d(B, E)+2d(CF, E)) = \frac{8.0623 + 2\times 7.2798}{3} \approx 7.5406$$

$$\begin{matrix} A,G \\ B,CF \\ D,H \\ E \end{matrix} \begin{bmatrix} 0. & & & \\ \boxed{4.8288} & 0. & & \\ 4.6436 & \boxed{6.3137} & 0. & \\ 7.2963 & \boxed{7.5406} & 11.0175 & 0. \end{bmatrix}.$$

Step 5: Merge AG,DH; calculate new distances and find the min. dist. to be $d(AGDH, BCF) = 5.5713$

$$d(AGDH, BCF) = \frac{1}{4\times 3}(6d(AG, BCF)+6d(BCF, DH)) = \frac{4.8288 + 6.3137}{2} \approx 5.5713$$

$$d(AGDH, E) = \frac{1}{4\times 1}(2d(AG, E) + 2d(DH, E)) = \frac{7.2963 + 11.0175}{2} \approx 9.1569$$

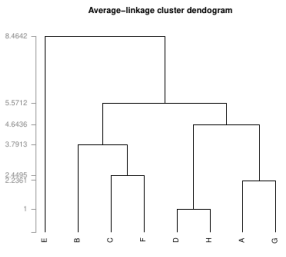
$$\begin{matrix} AG,DH \\ B,CF \\ E \end{matrix} \begin{bmatrix} 0. & & \\ \boxed{5.5713} & 0. & \\ \boxed{9.1569} & 7.5406 & 0. \end{bmatrix}.$$

Step 6: Merge AGDH, BCF; calculate new distances and find the min. dist. to be $d(,) =$

$$d(AGDHBCF, E) = \frac{1}{7\times 1}(4d(AGDH, E)+3d(BCF, E)) = \frac{4\times 9.1569 + 3\times 7.5406}{7} \approx 8.4642$$

$$\begin{matrix} AGDH,BCF \\ E \end{matrix} \begin{bmatrix} 0. & \\ \boxed{8.4642} & 0. \end{bmatrix}.$$

$$\begin{bmatrix} 3. & 7. & 1. & 2. \\ 0. & 6. & 2.2361 & 2. \\ 2. & 5. & 2.4495 & 2. \\ 1. & 10. & 3.7913 & 3. \\ 8. & 9. & 4.6436 & 4. \\ 11. & 12. & 5.5712 & 7. \\ 4. & 13. & 8.4642 & 8. \end{bmatrix}$$



Example 9.7.10 (May 2023 Semester Final Exam, Q5(b))**.**

Given the unlabelled data in Table 5.2.

Obs.	x_1	x_2	x_3
A	4.47	4.71	3.74
B	3.86	1.89	5.21
C	5.27	4.72	2.25
D	3.21	4.30	2.42
E	4.27	3.41	3.79

Table 5.2: Unlabelled data.

Suppose only part of (Euclidean) distance matrix of Table 5.2 is given:

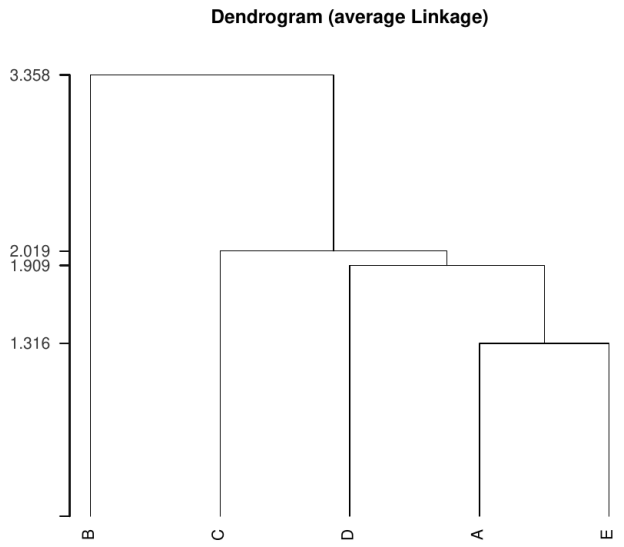
```
        A       B       C       D
A   0
B   3.238   0
C   1.691   4.331   0
D   1.870   3.744   2.109   0
```

1. Complete the distance matrix to include the point E in the distance matrix by rounding your calculations to 3 decimal places. (3 marks)

2. Construct a hierarchical clustering with **Euclidean distance** and **average linkage** using the results from part (i) and round all your calculations to at least 3 decimal places. (7 marks)

3. Draw the **dendrogram** of the hierarchical clustering in part (ii). (2 marks)

Solution: The dendrogram is shown below:



Marks are deducted for poor labelling or terribly drawn lines ..... [2 marks]

Other linkages are normally difficult to calculate manually and will not be explored.

9.7.3 BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is used to perform hierarchical clustering over particularly large data-sets. With modifications it can also be used to accelerate k-means clustering and Gaussian mixture modeling with the expectation–maximization algorithm. An advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points in an attempt to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database.

```
# R
stream::DSC_BIRCH(formula = NULL, threshold, branching,
  maxLeaf, maxMem = 0, outlierThreshold = 0.25)

# stream.data = DSD_Gaussians(k = 3, d = 2)
# BIRCH = DSC_BIRCH(threshold = .1, branching = 8, maxLeaf = 20)
# update(BIRCH, stream, n = 500)
# plot(BIRCH, stream)
```

9.7.4 DIANA

In divisive hierarchical clustering, all data is in the same cluster initially. The largest cluster is split until every object is separate. Because there are $O(2^n)$ ways of splitting each cluster,

heuristics are needed. DIANA (DIvisive ANAlysis) chooses the object with the maximum average dissimilarity and then moves all objects to this cluster that are more similar to the new cluster than to the remainder.

```
cluster::diana(x, diss = inherits(x, "dist"), metric = "euclidean",
  stand = FALSE, stop.at.k = FALSE,
  keep.diss = n < 100, keep.data = !diss, trace.lev = 0)
```