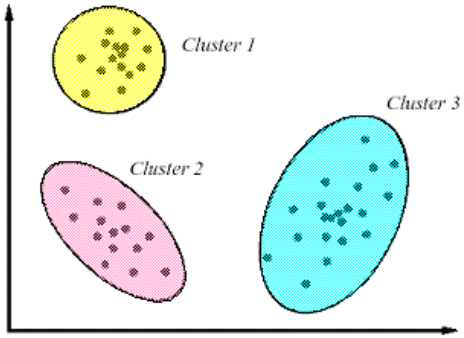
Cluster analysis and discriminant analysis

Cluster analysis

What is cluster analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters



- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

What is good clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on the similarity measure.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Measure the quality of clustering

- Dissimilarity/Similarity metric: Similarity is expressed in terms of a distance function, which is typically metric:

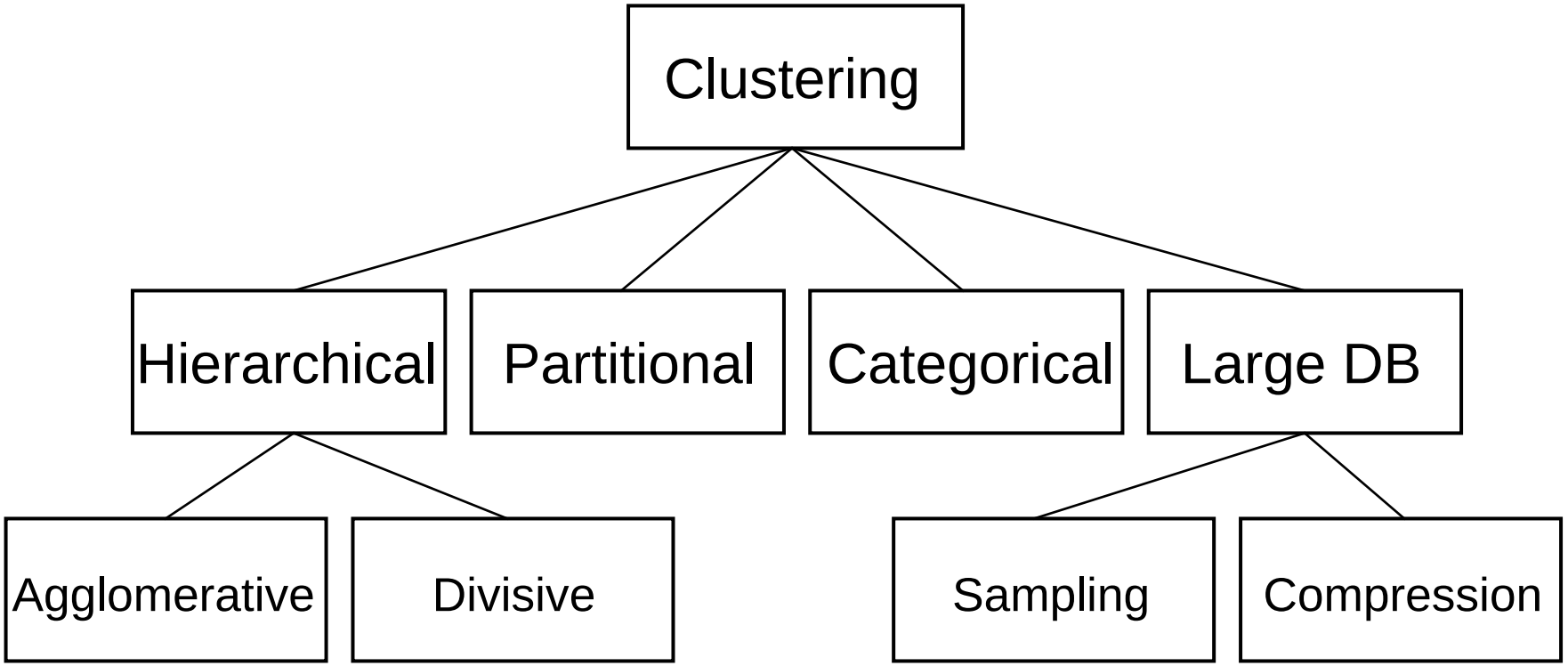
  $d(i, j)$

- The definitions of distance functions are usually very different for boolean, categorical, ordinal, interval-scaled, and ratio variables.

- Weights should be associated with different variables based on applications and data semantics.

- It is hard to define “similar enough” or “good enough”
  - the answer is typically highly subjective.

Clustering approaches



Data structures

- Data matrix

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- Dissimilarity matrix

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Partitioning algorithms: basic concept

- Partitioning method: construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
    - *k-means* (MacQueen 1967): Each cluster is represented by the center of the cluster
    - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw 1987): Each cluster is represented by one of the objects in the cluster

K-means clustering

- Basic ideas : using cluster centre (means) to represent cluster
- Assigning data elements to the closet cluster (centre).
- Goal: Minimise square error (intra-class dissimilarity):

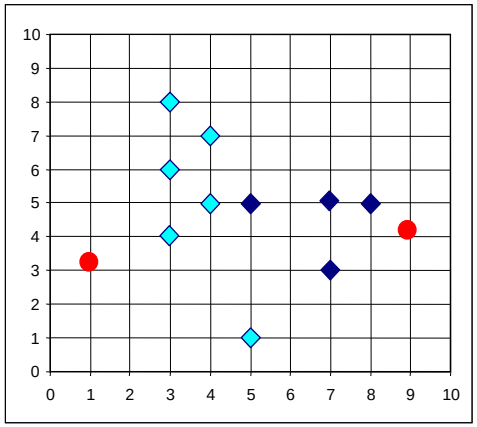
$$\sum_i d(\vec{x}_i, C(\vec{x}))$$

The K-means clustering method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster) $\vec{C}(S) = \sum_{i=1}^{n} \vec{X}_i / n, \vec{X}_1, \ldots, \vec{X}_n \in S$
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when no more new assignment

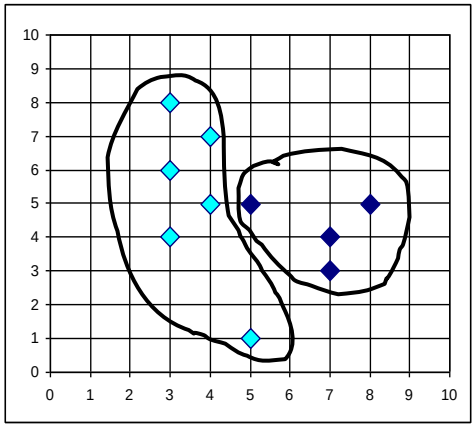
The K-means clustering method

Example

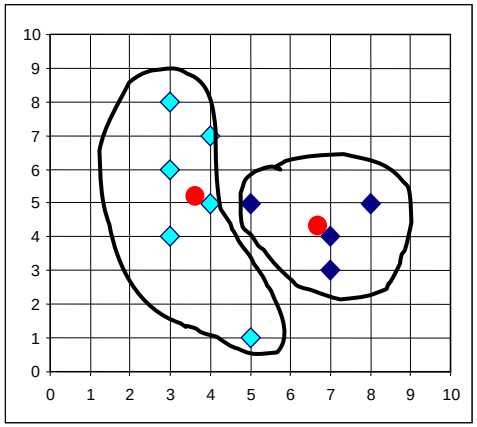


K=2

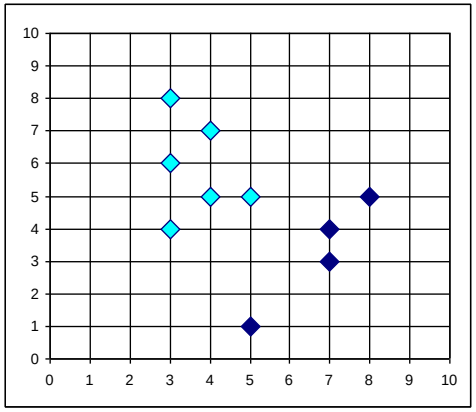
Arbitrarily choose K object as initial cluster center



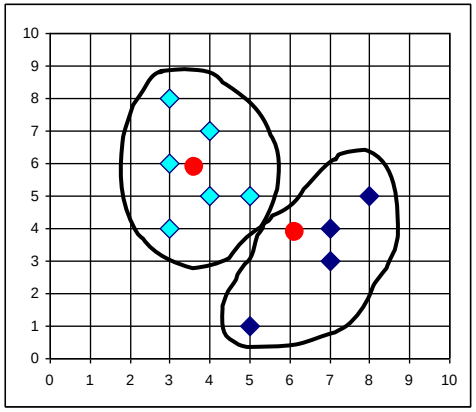
Assign each objects to most similar center



Update the cluster means



reassign



Update the cluster means

reassign

From Han and Kamber's slides on cluster analysis

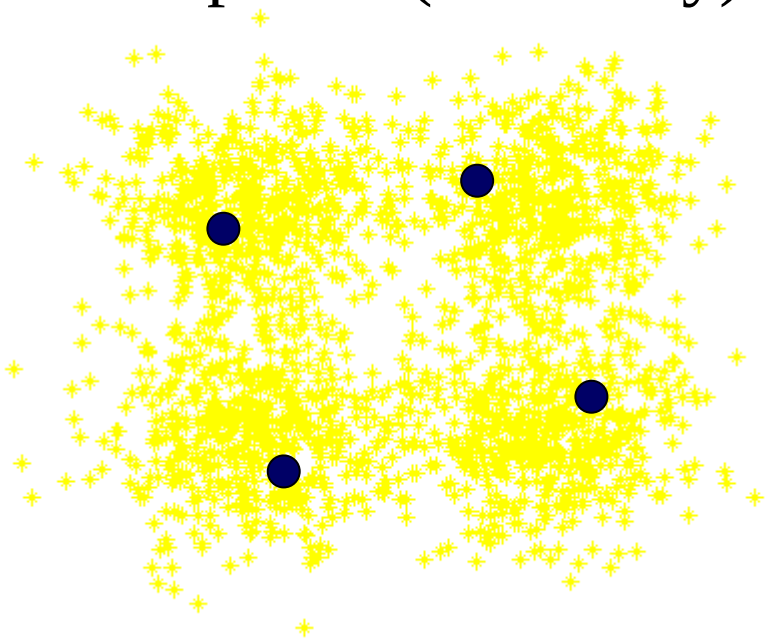
k-means Clustering : Procedure (1)

Initialization 1

Specify the number of cluster k :

for example, $k = 4$

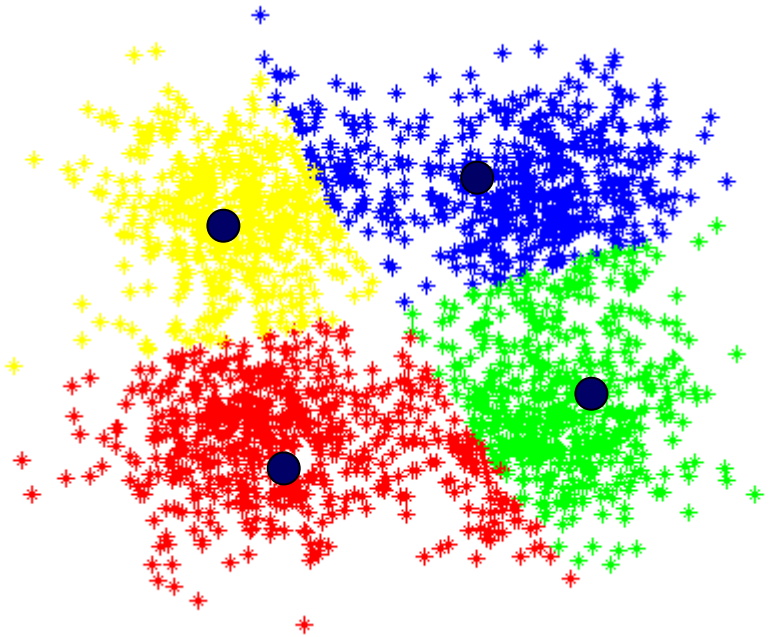
Select 4 points (randomly)



From Han and Kamber's slides on cluster analysis

k-means Clustering : Procedure (2)

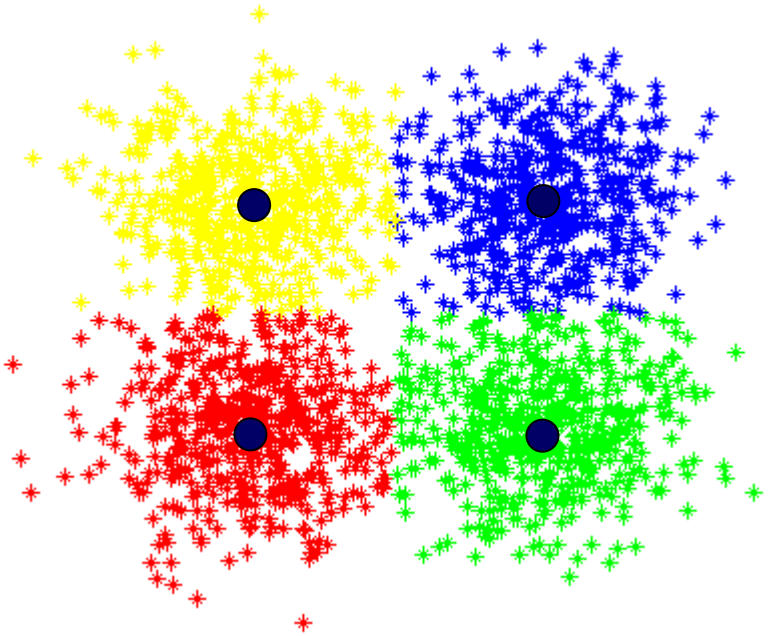
Each point is assigned to the nearest cluster based on the 4 distances



From Han and Kamber's slides on cluster analysis

k-means Clustering : Procedure (3)

Iterate until the means are converged



From Han and Kamber's slides on cluster analysis

R code

```
library(car)
mydata = DavisThin
tail(mydata)
```

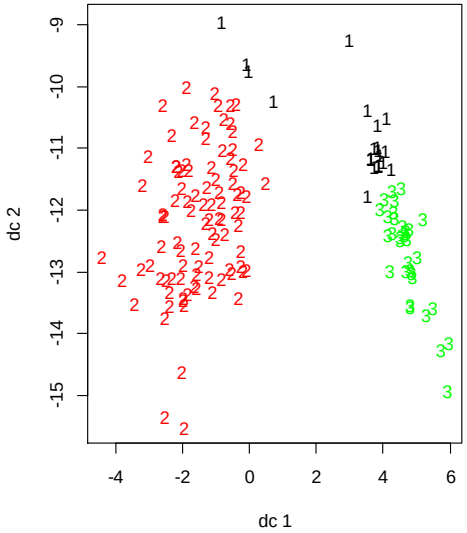
	DT1	DT2	DT3	DT4	DT5	DT6	DT7
186	1	1	0	0	0	0	0
187	0	0	0	0	0	0	0
188	2	0	1	0	3	0	1
189	0	0	0	0	2	0	0
190	0	0	0	0	1	3	0
191	0	0	0	0	0	0	0

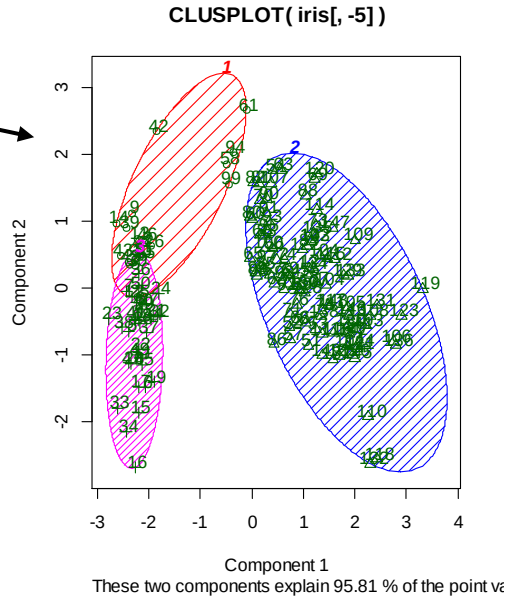
```
mydata = na.omit(mydata)
mydata = scale(mydata) # standardization

fit = kmeans(mydata, 4) # four clusters
table(fit$cluster)
output = cbind(mydata, type = fit$cluster); tail(output)
```

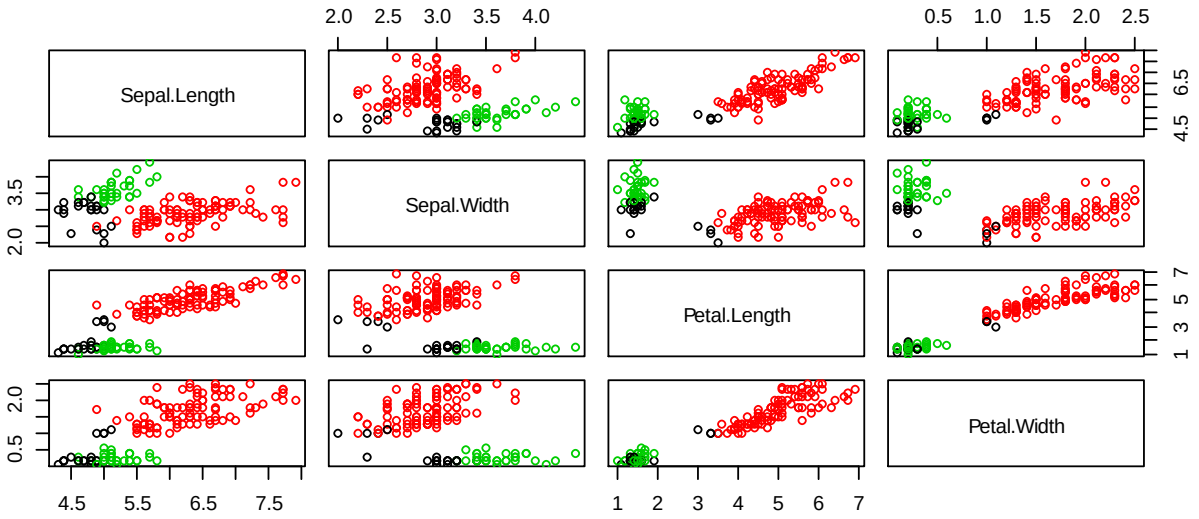
	DT1	DT2	DT3	DT4	DT5	DT6	DT7	type
186	0.637	-0.016	-0.765	-0.423	-0.841	-0.752	-0.537	1
187	-0.556	-0.797	-0.765	-0.423	-0.841	-0.752	-0.537	1
188	1.830	-0.797	0.033	-0.423	1.433	-0.752	0.412	3
189	-0.556	-0.797	-0.765	-0.423	0.675	-0.752	-0.537	1
190	-0.556	-0.797	-0.765	-0.423	-0.083	1.668	-0.537	3
191	-0.556	-0.797	-0.765	-0.423	-0.841	-0.752	-0.537	1

Plot k-means clustering

```
library(cluster); library(fpc)
data(iris)
# Kmeans clustre analysis
clus <- kmeans(iris[, -5], centers=3)

plotcluster(iris[, -5], clus$cluster)

clusplot(iris[, -5], clus$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0)

with(iris, pairs(iris[, -5], col=c(1:3)[clus$cluster]))
```

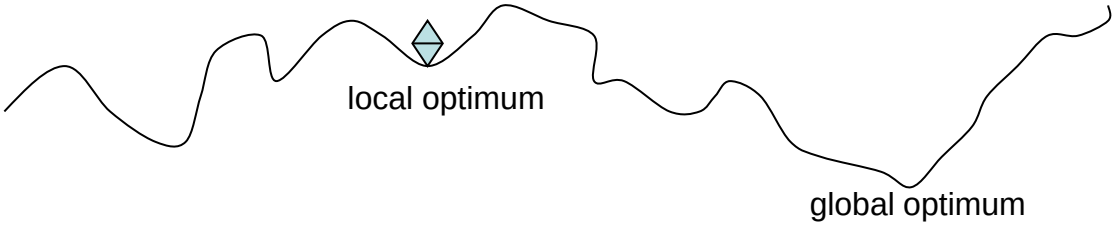






Comments on the K-means Method

- Strength: *relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
- Comment: often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*



- Weakness
  - Applicable only when *mean* is defined, not applicable to categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Methods of clustering similar to the K-means

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang 1998)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: k-prototype method

k-modes method

```
require(klaR)
X = sample(c('a','b','c'), 60, rep=T); X = matrix(X, ncol = 6)
colnames(X) = c('v1','v2','v3','v4','v5','v6')
cl <- kmodes(X, 3)
X = cbind(X, Cluster=cl$cluster)
X
```

v1	v2	v3	v4	v5	v6	Cluster
a	a	a	b	b	b	2
a	c	a	b	b	a	2
a	a	c	b	b	b	2
c	c	b	c	c	c	1
a	a	b	b	c	a	2
b	a	c	c	b	c	1
c	a	b	c	b	a	1
a	b	a	b	c	c	3
a	b	a	b	b	c	3
a	b	a	b	b	a	2

A simple-matching distance is used to determine the dissimilarity of two objects. It is computed by counting the number of mismatches in all variables. Alternative this distance is weighted by the frequencies of the categories in data (see Huang, 1997, for details).

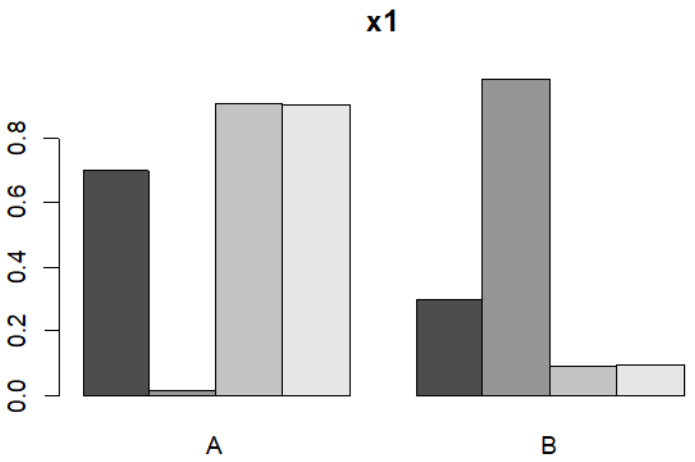
k-prototype method
(mixture of categorical and numerical data)

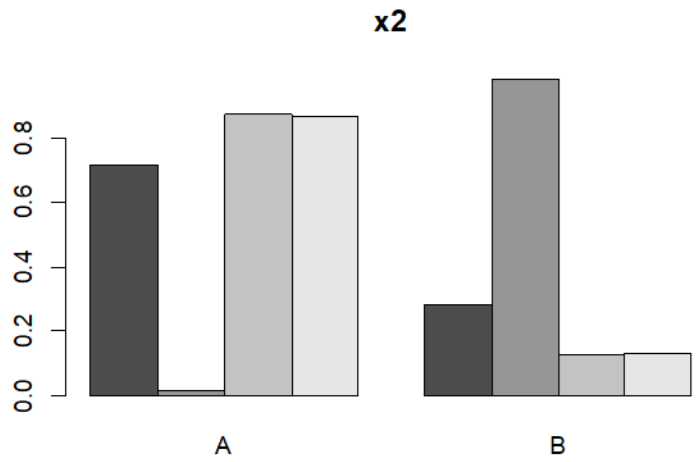
```
library(clustMixType)
# generate toy data with factors and numerics
n <- 100; prb <- 0.9; muk <- 1.5
clusid <- rep(1:4, each = n)
x1 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x1 <- c(x1, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb))); x1 <- as.factor(x1)
x2 <- sample(c("A","B"), 2*n, replace = TRUE, prob = c(prb, 1-prb))
x2 <- c(x2, sample(c("A","B"), 2*n, replace = TRUE, prob = c(1-prb, prb))); x2 <- as.factor(x2)
x3 <- c(rnorm (n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x4 <- c(rnorm(n, mean = -muk), rnorm(n, mean = muk), rnorm(n, mean = -muk), rnorm(n, mean = muk))
x <- data.frame(x1,x2,x3,x4)
# apply k prototyps
kpres <- kproto(x, 4)
X = cbind(Cluster = kpres$cluster, x)
par(mfrow=c(2,2))
clprofiles(kpres, x)
```

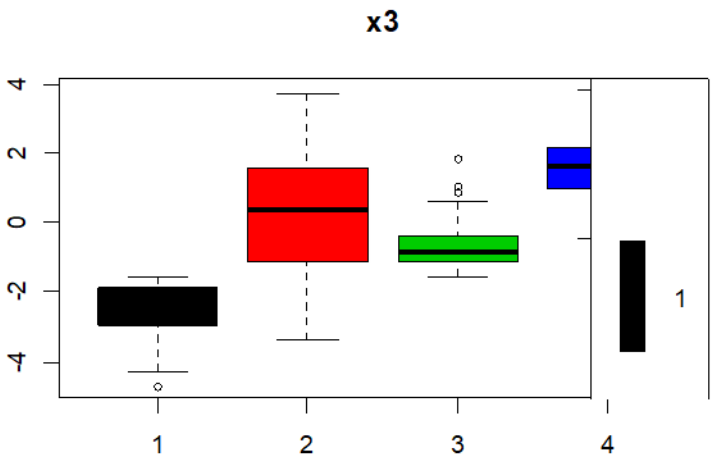
Cluster	x1	x2	x3	x4
1	A	A	0.278	0.323
2	B	B	2.275	0.479
2	B	B	1.776	1.120
2	B	B	1.696	1.403
2	B	B	3.480	2.426
2	B	B	2.713	0.977

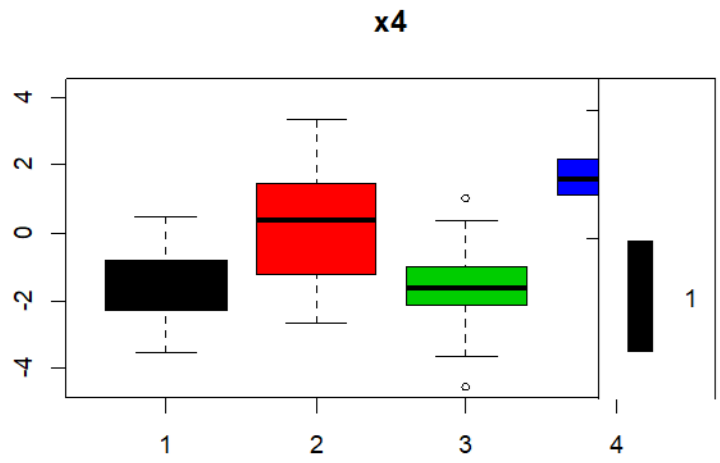
```
# in real world clusters are often not as clear cut
# by variation of lambda the emphasize is shifted towards factor / numeric variables
kpres <- kproto(x, 2, lambda = 0.1); clprofiles(kpres, x)
kpres <- kproto(x, 2, lambda = 25) ; clprofiles(kpres, x)
```

kpres <- kproto(x, 4); clprofiles(kpres, x)



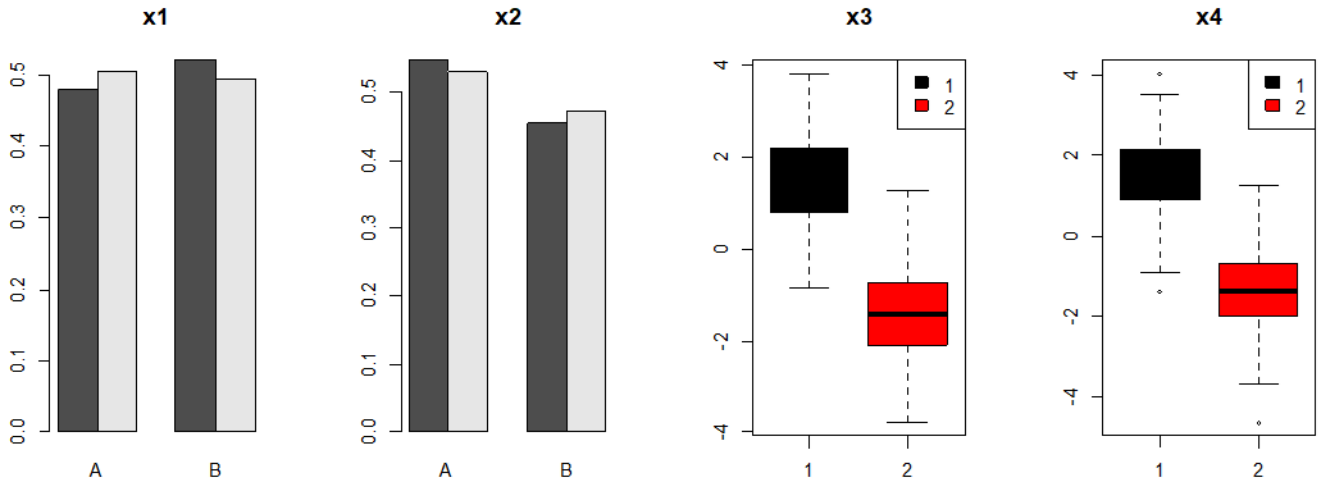


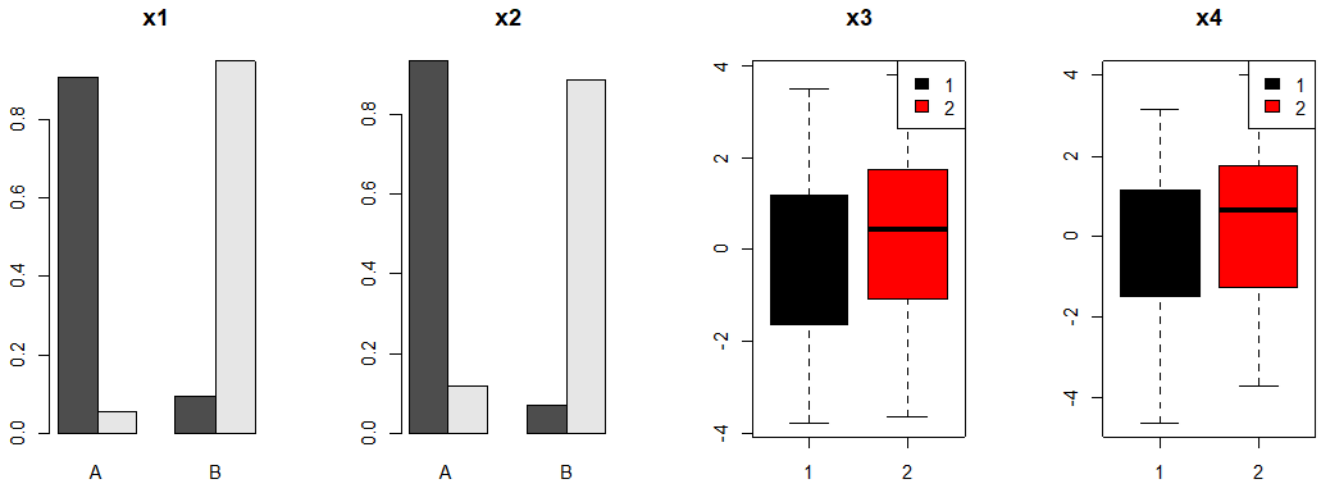




k-prototype method

kpres <- kproto(x, 2, lambda = 0.1); clprofiles(kpres, x)
kpres <- kproto(x, 2, lambda = 25); clprofiles(kpres, x)





Cluster	x1	x2	x3	x4
1	A	A	0.278	0.323
2	B	B	2.275	0.479
2	B	B	1.776	1.120
2	B	B	1.696	1.403
2	B	B	3.480	2.426
2	B	B	2.713	0.977

The K-medoids Clustering Method

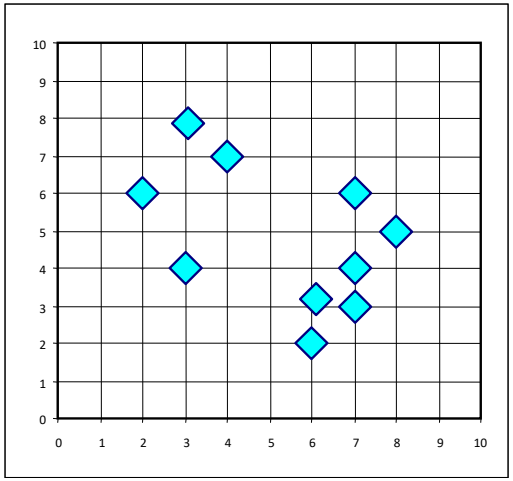
Find representative objects, called medoids, in clusters

- PAM (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - PAM works effectively for small data sets, but does not scale well for large data sets
- CLARA (Kaufmann & Rousseeuw, 1990)
- CLARANS (Ng & Han, 1994): randomized sampling

PAM (Partitioning Around Medoids) (1987)

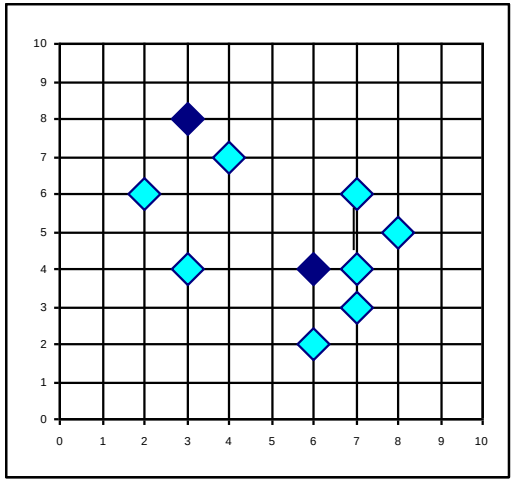
- PAM (Kaufman and Rousseeuw, 1987), built in Splus
- Use real object to represent the cluster
  1. Select ***k*** representative objects arbitrarily
  2. For each pair of non-selected object ***h*** and selected object ***i***, calculate the total swapping cost TC_{ih}
  3. For each pair of ***i*** and ***h***,
     - ✓If $TC_{ih} < 0$, ***i*** is replaced by ***h***
     - ✓Then assign each non-selected object to the most similar representative object
  4. repeat steps 2-3 until there is no change

Typical k-medoids algorithm (PAM)

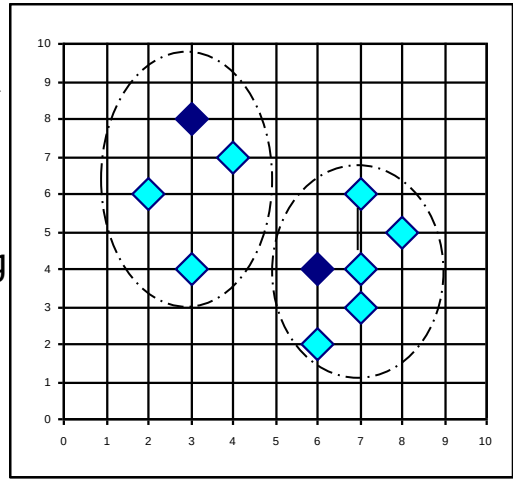


K=2

Arbitrary choose k object as initial medoids

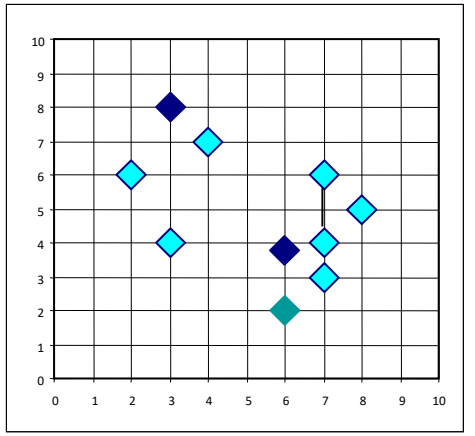


Assign each remaining object to nearest medoids

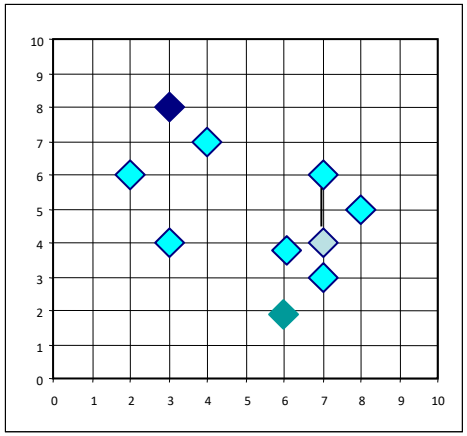


Total Cost = 20

Randomly select a nonmedoid object, O_{ramdom}



Compute total cost of swapping



Total Cost = 26

Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

From Han and Kamber's slides on cluster analysis

Comments on PAM?

- PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean

- PAM works efficiently for small data sets but does not **scale well** for large data sets.
  - $O(k(n-k)^2)$ for each iteration

    where n is # of data,k is # of clusters

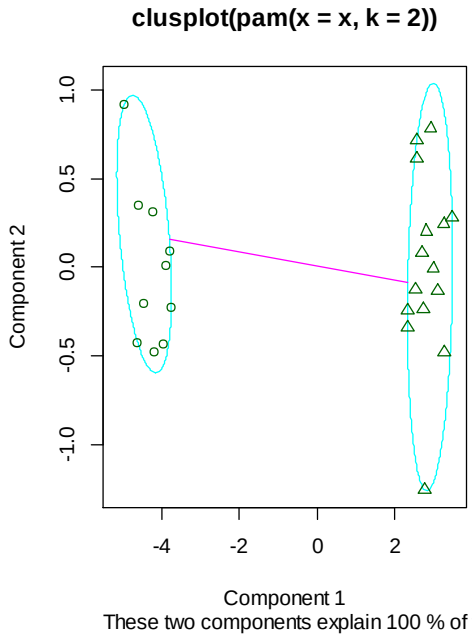
R code

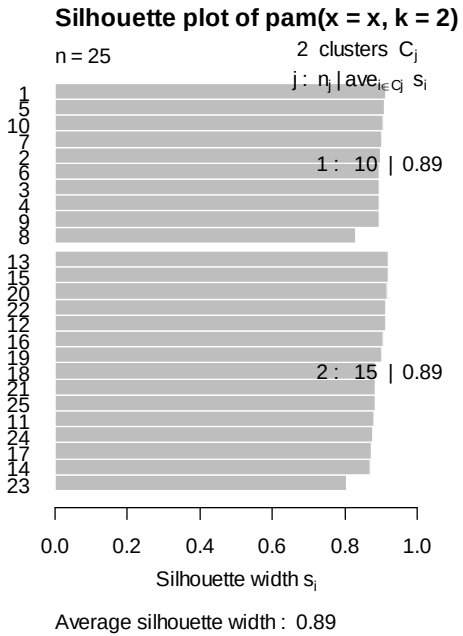
```
library(cluster)
## generate 25 objects, divided into 2 clusters.
x <- rbind(cbind(rnorm(10,0,0.5), rnorm(10,0,0.5)),
           cbind(rnorm(15,5,0.5), rnorm(15,5,0.5)))
pamx <- pam(x, 2)
pamx$medoids
pamx$clustering
summary(pamx)
par(mfrow=c(1,2))
plot(pamx)
```

```
          [,1]      [,2]
[1,] 0.4711244 0.1125793
[2,] 4.8384212 5.0801553
```

```
[1] 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
```

	[,1]	[,2]
[1,]	0.2401	-0.4645
[2,]	0.7500	-0.2493
[3,]	-0.2663	-0.2042
[4,]	0.4903	0.2477
[5,]	0.4711	0.1126
[6,]	0.3084	-0.7229
[7,]	0.6263	-0.4478
[8,]	-0.9210	-0.0823
[9,]	0.7276	0.0358
[10,]	0.0161	0.0548
[11,]	4.3500	5.3324
[12,]	5.0964	4.8931
[13,]	5.0869	5.2267
[14,]	4.2643	5.3940
[15,]	4.8384	5.0802
[16,]	4.8669	4.8043
[17,]	4.4566	5.7081
[18,]	5.1997	5.7838
[19,]	5.0837	5.5980
[20,]	4.8168	5.2423
[21,]	5.6312	5.1233
[22,]	5.2558	5.2271
[23,]	5.8675	4.2283
[24,]	4.8923	4.5124
[25,]	4.8244	4.5802



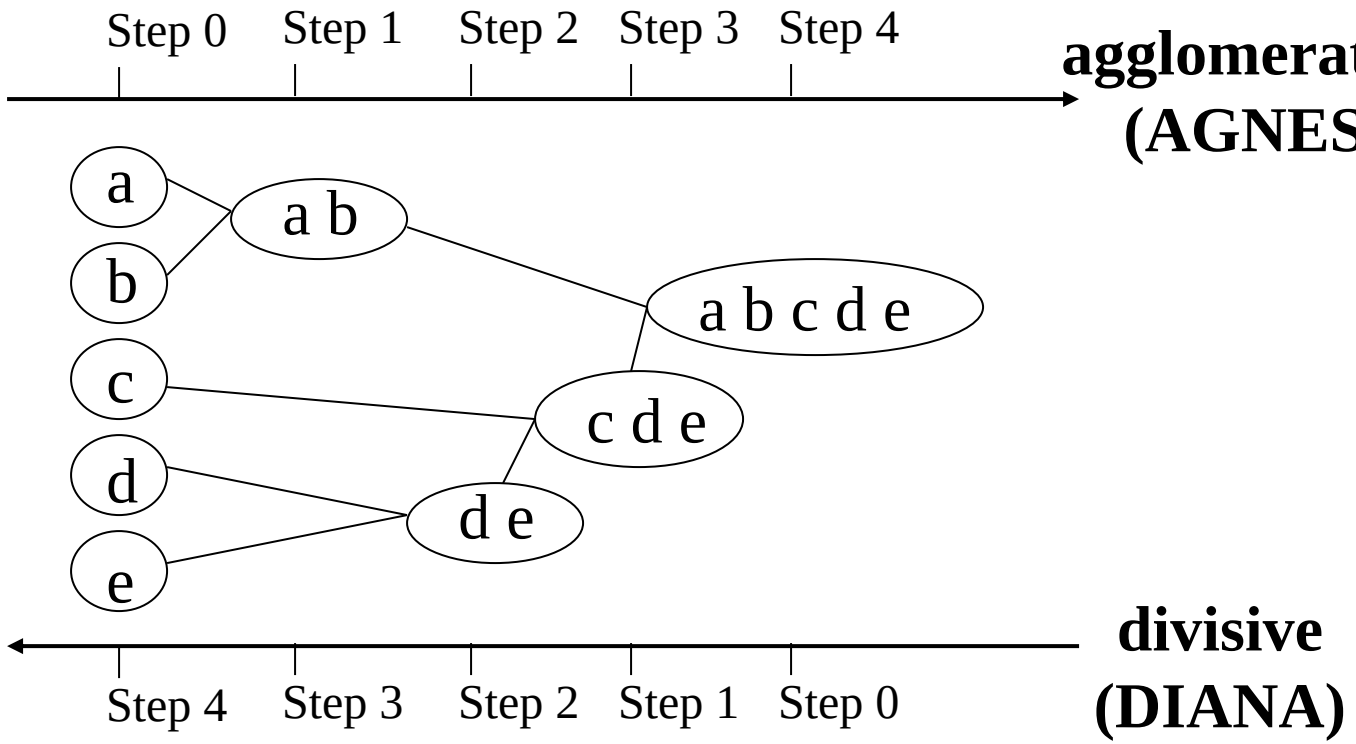


CLARA (Clustering Large Applications) (1990)

- CLARA (Kaufmann and Rousseeuw in 1990)
  - Built in statistical analysis packages, such as S+
- It draws multiple samples of the data set, applies *PAM* on each sample, and gives the best clustering as the output
- Strength: deals with larger data sets than *PAM*
- Weakness:
  - Efficiency depends on the sample size
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased

Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



From Han and Kamber's slides on cluster analysis

Hierarchical Clustering

Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the basic process hierarchical clustering is this:

1. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item.

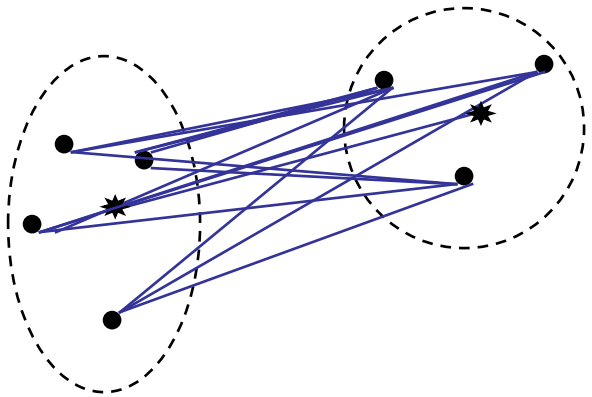
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.

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.

Distance between clusters

- ***Single Link***: smallest distance between points
- ***Complete Link:*** largest distance between points
- ***Average Link:*** average distance between points
- ***Centroid:*** distance between centroids



Amalgamation or linkage rules

Single linkage (nearest neighbor). The distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters. This rule will, in a sense, *string* objects together to form clusters, and the resulting clusters tend to represent long "chains."

Complete linkage (furthest neighbor). The distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors"). This method usually performs quite well in cases when the objects actually form naturally distinct "clumps." If the clusters tend to be somehow elongated or of a "chain" type nature, then this method is inappropriate.

Unweighted pair-group average. The distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters. This method is also very efficient when the objects form natural distinct "clumps," however, it performs equally well with elongated, "chain" type clusters. Note that in their book, Sneath and Sokal (1973) introduced the abbreviation UPGMA to refer to this method as *unweighted pair-group method using arithmetic averages*.

Weighted pair-group average. This method is identical to the *unweighted pair-group average* method, except that in the computations, the size of the respective clusters (i.e., the number of objects contained in them) is used as a weight. Thus, this method (rather than the previous method) should be used when the cluster sizes are suspected to be greatly uneven. Note that in their book, Sneath and Sokal (1973) introduced the abbreviation *WPGMA* to refer to this method as *weighted pair-group method using arithmetic averages*.

Unweighted pair-group centroid. The *centroid* of a cluster is the average point in the multidimensional space defined by the dimensions. In a sense, it is the *center of gravity* for the respective cluster. In this method, the distance between two clusters is determined as the difference between centroids. Sneath and Sokal (1973) use the abbreviation *UPGMC* to refer to this method as *unweighted pair-group method using the centroid average*.

Weighted pair-group centroid (median). This method is identical to the previous one, except that weighting is introduced into the computations to take into consideration differences in cluster sizes (i.e., the number of objects contained in them). Thus, when there are (or one suspects there to be) considerable differences in cluster sizes, this method is preferable to the previous one. Sneath and Sokal (1973) use the abbreviation *WPGMC* to refer to this method as *weighted pair-group method using the centroid average*.

Ward's method. This method is distinct from all other methods because it uses an analysis of variance approach to evaluate the distances between clusters. In short, this method attempts to minimize the Sum of Squares (SS) of any two (hypothetical) clusters that can be formed at each step. Refer to Ward (1963) for details concerning this method. In general, this method is regarded as very efficient, however, it tends to create clusters of small size.

Distance measures

Euclidean distance. This is probably the most commonly chosen type of distance. It simply is the geometric distance in the multidimensional space. It is computed as: distance(x,y) = $\{\Sigma_i (x_i - y_i)^2\}^{½}$

Note that Euclidean (and squared Euclidean) distances are usually computed from raw data, and not from standardized data. This method has certain advantages (e.g., the distance between any two objects is not affected by the addition of new objects to the analysis, which may be outliers). However, the distances can be greatly affected by differences in scale among the dimensions from which the distances are computed. For example, if one of the dimensions denotes a measured length in centimeters, and you then convert it to millimeters (by multiplying the values by 10), the resulting Euclidean or squared Euclidean distances (computed from multiple dimensions) can be greatly affected (i.e., biased by those dimensions which have a larger scale), and consequently, the results of cluster analyses may be very different. Generally, it is good practice to transform the dimensions so they have similar scales.

Squared Euclidean distance. You may want to square the standard Euclidean distance in order to place progressively greater weight on objects that are further apart. This distance is computed as (see also the note in the previous paragraph):

distance(x,y) = $\Sigma_i (x_i - y_i)^2$

City-block (Manhattan) distance. This distance is simply the average difference across dimensions. In most cases, this distance measure yields results similar to the simple Euclidean distance. However, note that in this measure, the effect of single large differences (outliers) is dampened (since they are not squared). The city-block distance is computed as:

distance(x,y) = $\Sigma_i |x_i - y_i|$

Chebychev distance. This distance measure may be appropriate in cases when one wants to define two objects as "different" if they are different on any one of the dimensions. The Chebychev distance is computed as:

distance(x,y) = Maximum$|x_i - y_i|$

Power distance. Sometimes one may want to increase or decrease the progressive weight that is placed on dimensions on which the respective objects are very different. This can be accomplished via the *power distance*. The power distance is computed as:

distance(x,y) = $(\Sigma_i |x_i - y_i|^p)^{1/r}$

where *r* and *p* are user-defined parameters. A few example calculations may demonstrate how this measure "behaves." Parameter *p* controls the progressive weight that is placed on differences on individual dimensions, parameter *r* controls the progressive weight that is placed on larger differences between objects. If *r* and *p* are equal to 2, then this distance is equal to the Euclidean distance.

Percent disagreement. This measure is particularly useful if the data for the dimensions included in the analysis are categorical in nature. This distance is computed as: distance(x,y) = (Number of $x_i \neq y_i$)/ i.

Distance Measures: Minkowski Metric

Suppose two objects x and y both have p features:

$$x = (x_1 x_2 \cdots x_p)$$

$$y = (y_1 y_2 \cdots y_p)$$

The Minkowski metric is defined by

$$d(x, y) = \sqrt[r]{\sum_{i=1}^{p} | x_i - y_i |^r}$$

Commonly Used Minkowski Metrics

$1, r = 2$ (Euclidean distance)

$$d(x, y) = \sqrt[2]{\sum_{i=1}^{p} | x_i - y_i |^2}$$

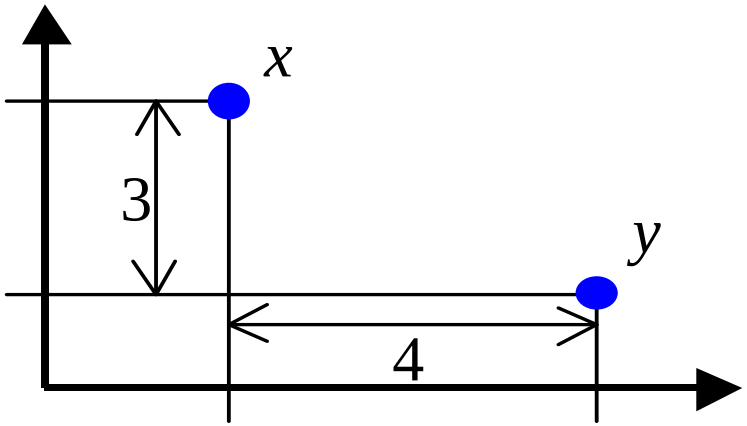
$2, r = 1$ (Manhattan distance)

$$d(x, y) = \sum_{i=1}^{p} | x_i - y_i |$$

$3, r = +\infty$ ("sup" distance)

$$d(x, y) = \max_{1 \le i \le p} | x_i - y_i |$$

An Example



1, Euclidean distance: $\sqrt[2]{4^2+3^2}=5.$

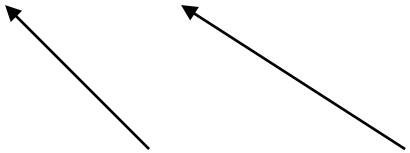
2, Manhattan distance: $4+3=7.$

3, "sup" distance: $\max\{4,3\}=4.$

Manhattan distance is called *Hamming distance* when all features are binary.

Gene expression levels under 17 conditions (1-High,0-Low)

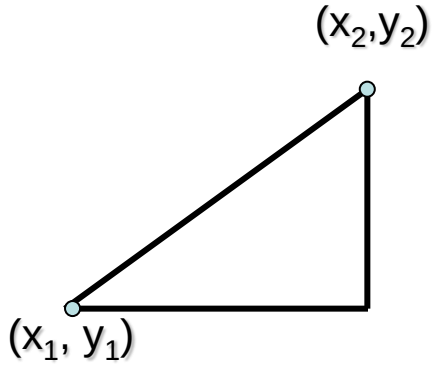
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
GeneA	0	1	1	0	0	1	0	0	1	0	0	1	1	1	0	0	1
GeneB	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1



Hamming Distance: $\#(01)+\#(10)=4+1=5$**.**

Other similarity indices

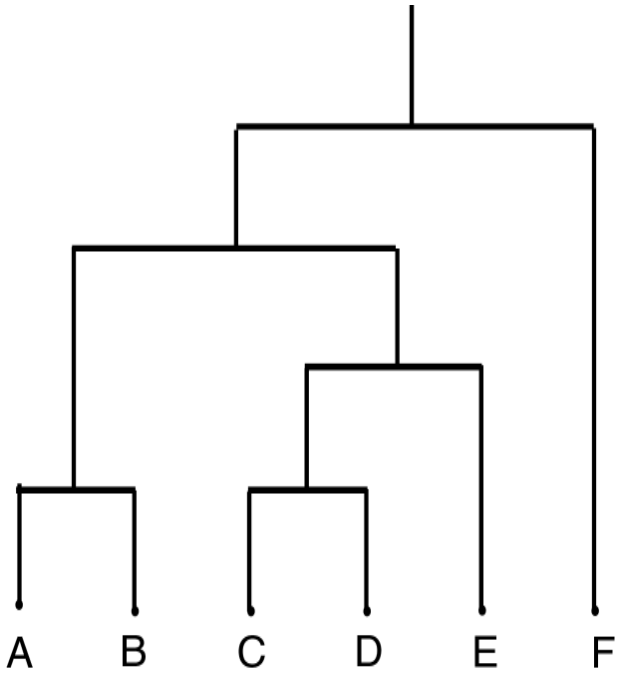
- L_m: $(|x_1-x_2|^m+|y_1-y_2|^m)^{1/m}$ (power distance)
- L_∞: $\max(|x_1-x_2|,|y_1-y_2|)$ (sup distance)
- Inner product: $x_1x_2+y_1y_2$
- Pearson correlation coefficient
- Spearman rank correlation coefficient



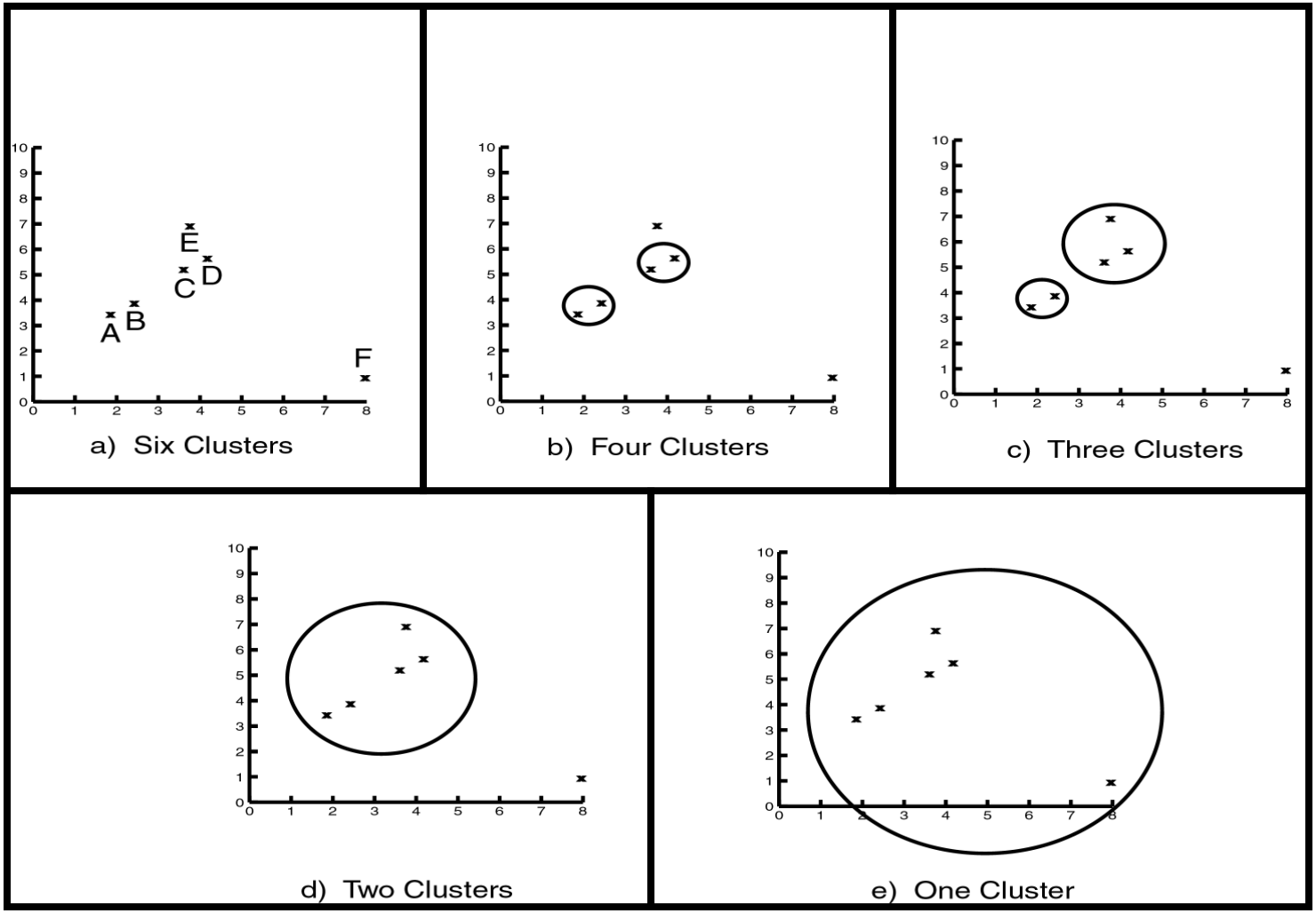
Dendrogram

A tree data structure which illustrates hierarchical clustering techniques.

- Each level shows clusters for that level.
  - Leaf – individual clusters
  - Root – one cluster
- A cluster at level i is the union of its children clusters at level i+1.

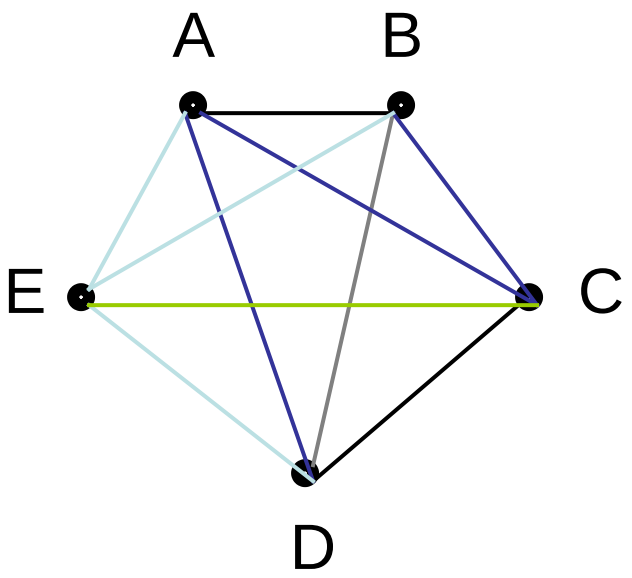


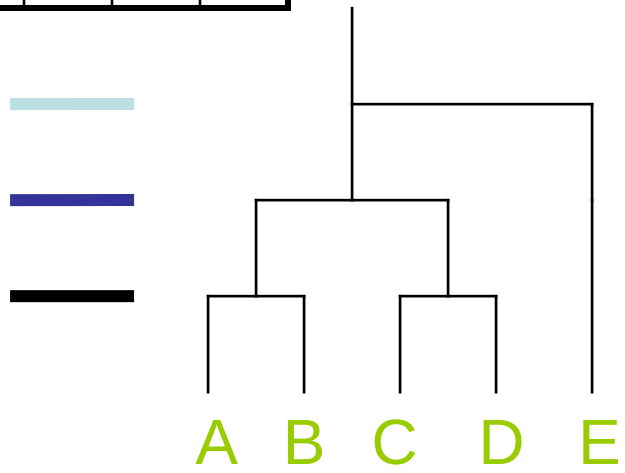
Levels of clustering



Agglomerative example

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

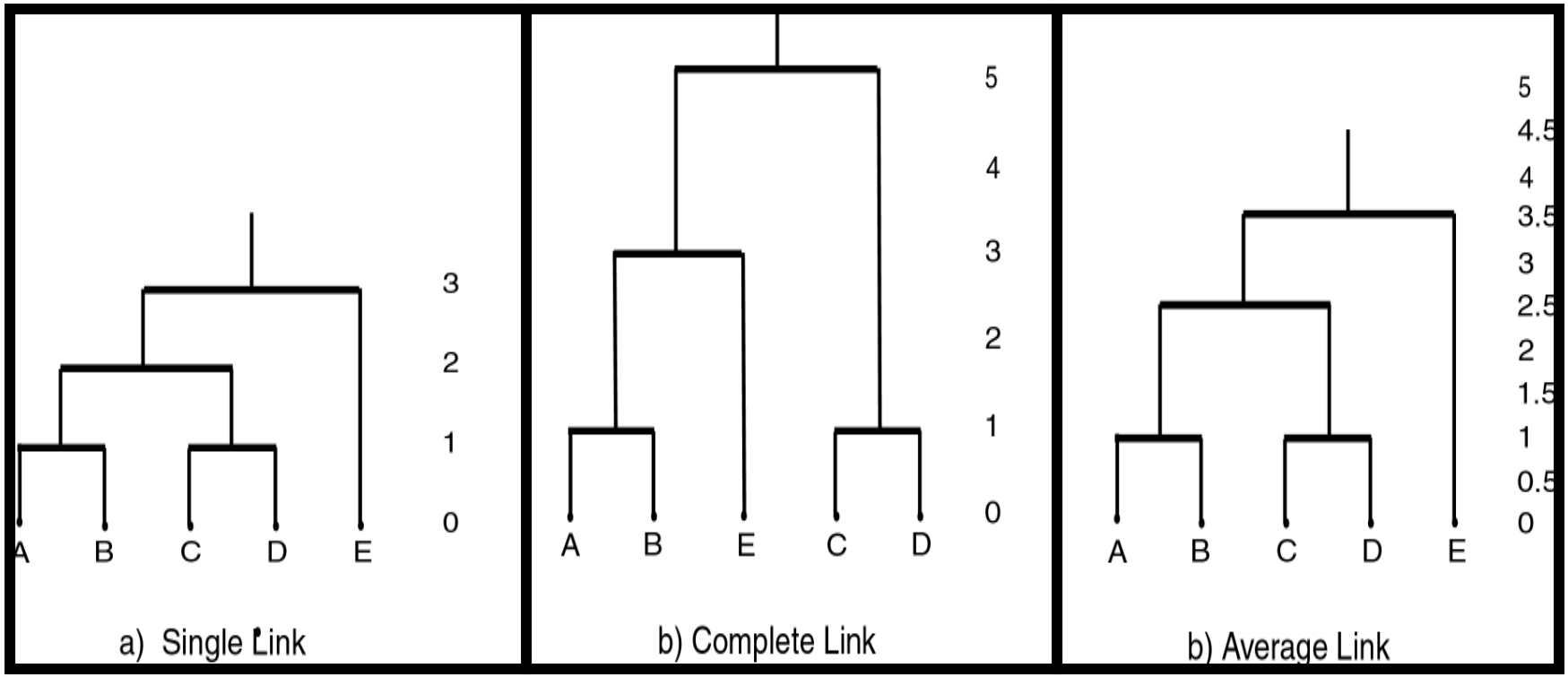




Threshold of

1 2 3 4 5

Single-Link, Complete-Link & Average-Link Clustering



Issues in cluster analysis

- A lot of clustering algorithms
- A lot of distance/similarity metrics
- Which clustering algorithm runs faster and uses less memory?
- How many clusters after all?
- Are the clusters stable?
- Are the clusters meaningful?

Statistical significance testing

- Cluster analysis is a "collection" of different algorithms that "put objects into clusters according to well defined similarity rules." not as much a typical statistical test

- Cluster analysis methods are mostly used when we do **not have any a priori hypotheses**, but are still in the exploratory phase of our research. In a sense, cluster analysis finds the "most significant solution possible."

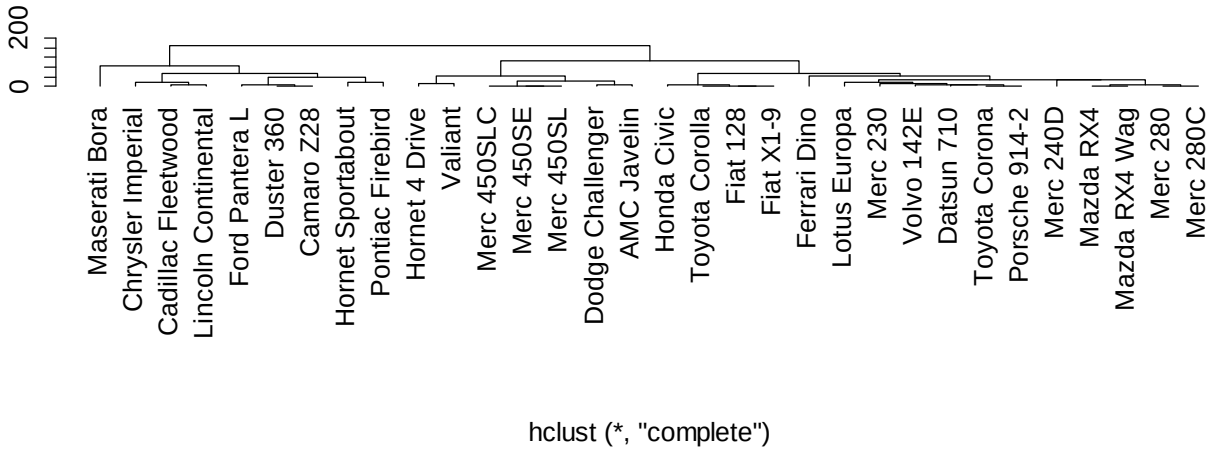
- Statistical significance testing is not appropriate here, even in cases when p-levels are reported (as in *k*-means clustering).

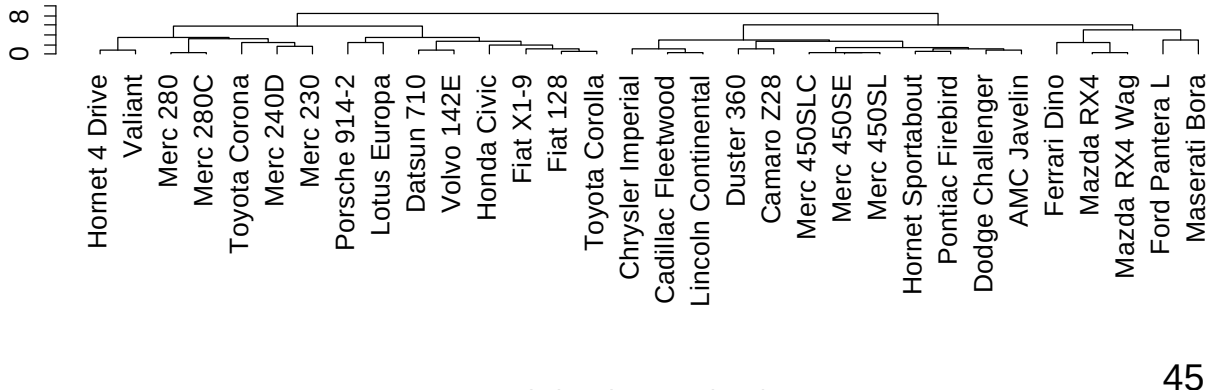
R code: Hierarchical cluster analysis

mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda_RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4
Mazda_RX4_Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun_710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet_4_Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet_Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

```
# Hierarchical cluster analysis

cluster1 = hclust(dist(mtcars))
# plot cluster dendrogram
plot(cluster1, hang = -1)

# standardize variables
cluster2 = hclust(dist(scale(mtcars)))
plot(cluster2, hang = -1)
```





Discriminant analysis (DA)

Discriminant analysis (DA)

- DA is used to identify boundaries between groups of objects by a measure of distance.

  For example:

  (a) What taxa do some insects belong to on basis of a number of measures.

  (b) Is someone a good credit risk or not?

  (c) Should a student be admitted to college?

- Similar to regression, except that criterion (or dependent variable) is categorical rather than continuous.
- Alternatively, discriminant function analysis is multivariate analysis of variance (MANOVA) **reversed**.

  In MANOVA, the independent variables are the groups and the dependent variables are the continuous measures. In DA, the independent variables are the continuous measures and the dependent variables are the groups.

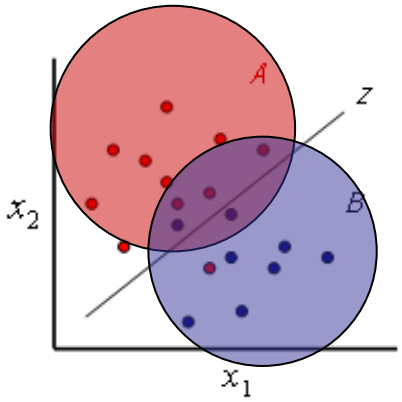
Example

Discriminant analysis of remote sensing data on five crops

crop	band1	band2	band3	band4
CORN	16	27	31	33
CORN	15	23	30	30
CORN	16	27	27	26
CORN	18	20	25	23
CORN	15	15	31	32
CORN	15	32	32	15
CORN	12	15	16	73
SOYBEANS	20	23	23	25
SOYBEANS	24	24	25	32
SOYBEANS	21	25	23	24
SOYBEANS	27	45	24	12
SOYBEANS	12	13	15	42
SOYBEANS	22	32	31	43
COTTON	31	32	33	34
COTTON	29	24	26	28
COTTON	34	32	28	45
COTTON	26	25	23	24
COTTON	53	48	75	26
COTTON	34	35	25	78
SUGARBEETS	22	23	25	42
SUGARBEETS	25	25	24	26
SUGARBEETS	34	25	16	52
SUGARBEETS	54	23	21	54
SUGARBEETS	25	43	32	15
SUGARBEETS	26	54	2	54
CLOVER	12	45	32	54
CLOVER	24	58	25	34
CLOVER	87	54	61	21
CLOVER	51	31	31	16
CLOVER	96	48	54	62
CLOVER	31	31	11	11
CLOVER	56	13	13	71
CLOVER	32	13	27	32
CLOVER	36	26	54	32
CLOVER	53	08	06	54
CLOVER	32	32	62	16

Linear discriminant analysis

- Linear discriminant analysis attempts to find the linear combination of the selected measures that best separate the population



$$Z = b_1 x_1 + b_2 x_2$$

b = discriminant coefficients
x = input variables

Procedure

Discriminant function analysis is broken into a 2-step process:

1. testing significance of a set of discriminant functions

The first step is computationally identical to MANOVA. There is a matrix of total variances and covariances; likewise, there is a matrix of pooled within-group variances and covariances.

The two matrices are compared via multivariate F tests in order to determine whether or not there are any significant differences (with regard to all variables) between groups.

One first performs the multivariate test, and, if statistically significant, proceeds to see which of the variables have significantly different means across the groups.

Procedure

2. classification

- Once group means are found to be statistically significant, classification of variables is undertaken.
- Discriminant analysis automatically determines some optimal combination of variables so that the first function provides the most overall discrimination between groups, the second provides second most, and so on.
- Moreover, the functions will be independent or orthogonal, that is, their contributions to the discrimination between groups will not overlap.

Assumptions

Sample size:
Unequal sample sizes are acceptable. The sample size of the smallest group needs to exceed the number of predictor variables. As a “rule of thumb”, the smallest sample size should be at least 20 for a few (4 or 5) predictors. The maximum number of independent variables is n - 2, where n is the sample size. While this low sample size may work, it is not encouraged, and generally it is best to have 4 or 5 times as many observations and independent variables.

Normal distribution:
It is assumed that the data (for the variables) represent a sample from a multivariate normal distribution. You can examine whether or not variables are normally distributed with histograms of frequency distributions. However, note that violations of the normality assumption are not "fatal" and the resultant significance test are still reliable as long as non-normality is caused by skewness and not outliers (Tabachnick and Fidell 1996).

Homogeneity of variances/covariances:
Discriminant analysis is very sensitive to heterogeneity of variance-covariance matrices. Before accepting final conclusions for an important study, it is a good idea to review the within-groups variances and correlation matrices. Homoscedasticity is evaluated through scatterplots and corrected by transformation of variables.

Assumptions

Outliers:

- Discriminant analysis is highly sensitive to the inclusion of outliers.
- Run a test for univariate and multivariate outliers for each group, and transform or eliminate them.
- If one group in the study contains extreme outliers that impact the mean, they will also increase variability. Overall significance tests are based on pooled variances, that is, the average variance across all groups. Thus, the significance tests of the relatively larger means (with the large variances) would be based on the relatively smaller pooled variances, resulting erroneously in statistical significance.

Non-multicollinearity:

- If one of the independent variables is very highly correlated with another, or one is a function (e.g., the sum) of other independents, then the matrix will not have a unique discriminant solution.
- To the extent that independents are correlated, the standardized discriminant function coefficients will not reliably assess the relative importance of the predictor variables.

R code: linear discriminant analysis

```
library(MASS);  remote.sensing = read.csv("remote.sensing.csv", header = T);
table(remote.sensing$crop)
```

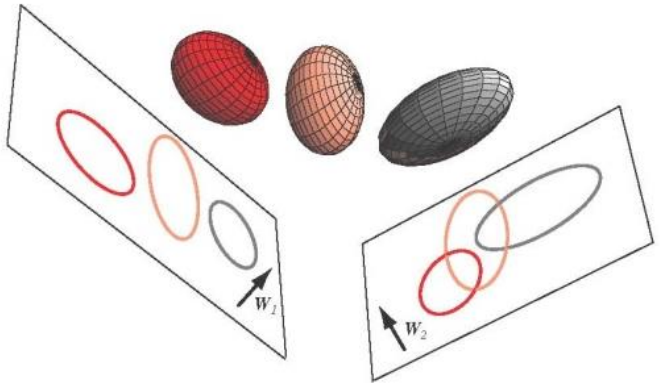
CLOVER	CORN	COTTON	SOYBEANS	SUGARBEETS
11	7	6	6	6

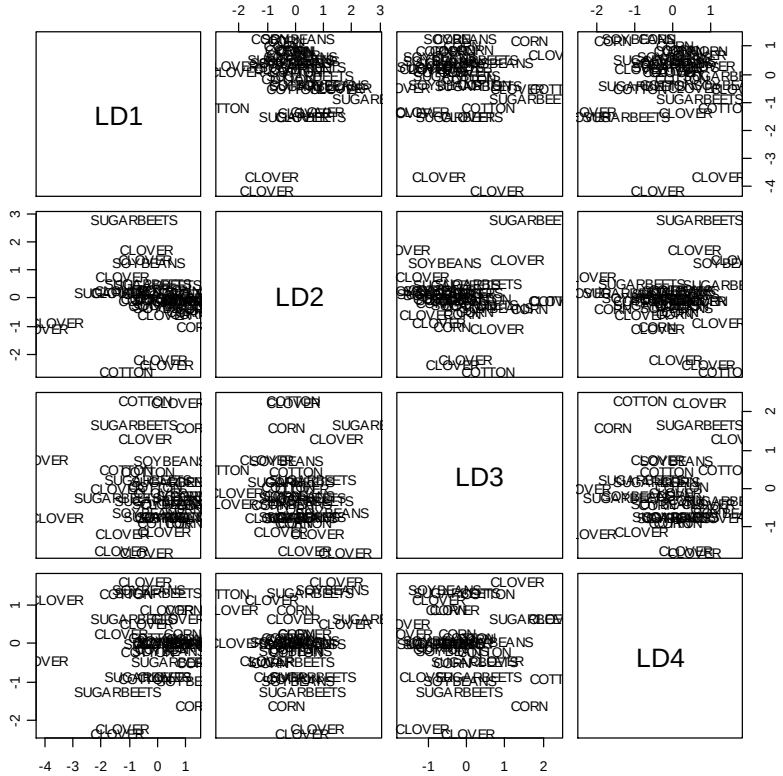
```
nrow(remote.sensing) # 36
```

```
lda.result <- lda(crop ~ band1+band2+band3+band4, remote.sensing)
```

```
plot(lda.result, cex = 1)
```

```
> ?lda
```





R: linear discriminant analysis

lda.result

```
Call:
lda(crop ~ band1 + band2 + band3 + band4, data = remote.sensing)

Prior probabilities of groups:
    CLOVER       CORN     COTTON   SOYBEANS SUGARBEETS
 0.3055556  0.1944444  0.1666667  0.1666667  0.1666667

Group means:
              band1    band2    band3    band4
CLOVER     46.36364 32.63636 34.18182 36.63636
CORN       15.28571 22.71429 27.42857 33.14286
COTTON     34.50000 32.66667 35.00000 39.16667
SOYBEANS   21.00000 27.00000 23.50000 29.66667
SUGARBEETS 31.00000 32.16667 20.00000 40.50000

Coefficients of linear discriminants:
                 LD1           LD2          LD3           LD4
band1 -6.147360e-02  0.009215431 -0.02987075 -0.014680566
band2 -2.548964e-02  0.042838972  0.04631489  0.054842132
band3  1.642126e-02 -0.079471595  0.01971222  0.008938745
band4  5.143616e-05 -0.013917423  0.05381787 -0.025717667

Proportion of trace:
   LD1    LD2    LD3    LD4
0.7364 0.1985 0.0576 0.0075
```

R: linear discriminant analysis

lda.predict <- predict(lda.result, remote.sensing)

table(lda.predict$class) ***#number of observation for each crop***

lda.predict$class ***#predicted crop types***

```
 [1] CORN      CORN      CORN      CORN      CORN      SOYBEANS
 [7] CORN      SOYBEANS  SOYBEANS  SOYBEANS  SUGARBEETS CORN
…
```

lda.predict$posterior ***#predicted crop types values***

	CLOVER	CORN	COTTON	SOYBEANS	SUGARBEETS
1	0.08935	0.4054295	0.17631	0.239184	0.08971
2	0.07690	0.4558027	0.14209	0.253010	0.07219

…

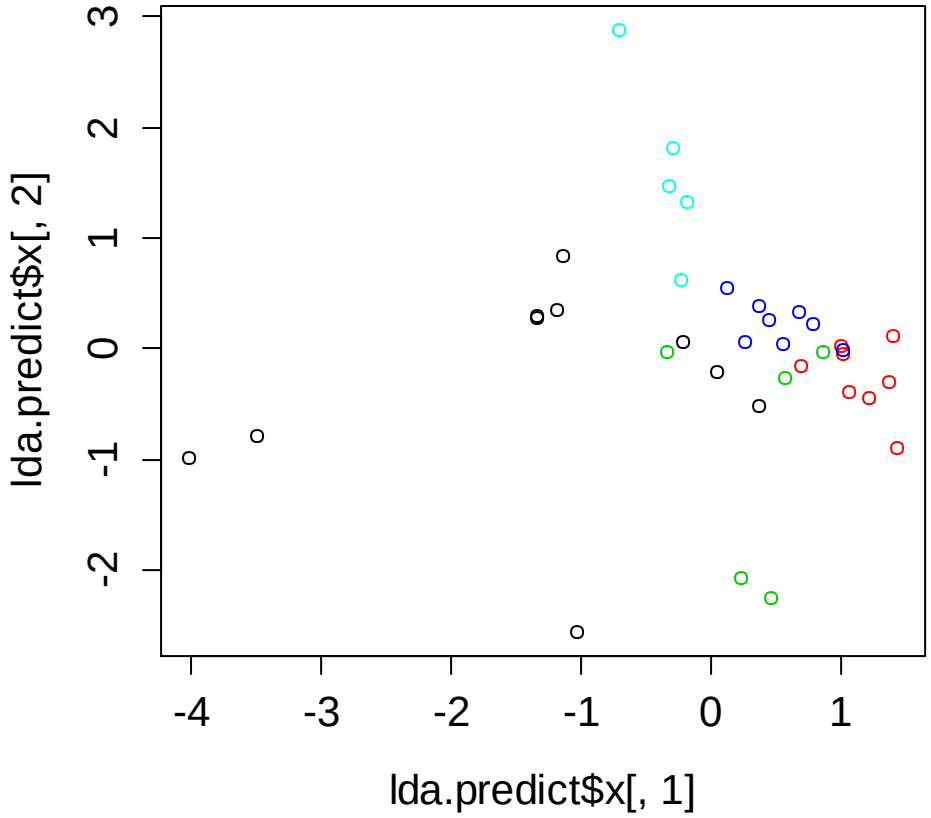
lda.predict$x ***#linear discriminant scores for each observation***

	LD1	LD2	LD3	LD4
1	1.05991249	-0.38894000	0.22804664	0.17329536
2	1.20676907	-0.44828745	-0.10850799	0.03682165

…

R - linear discriminant analysis

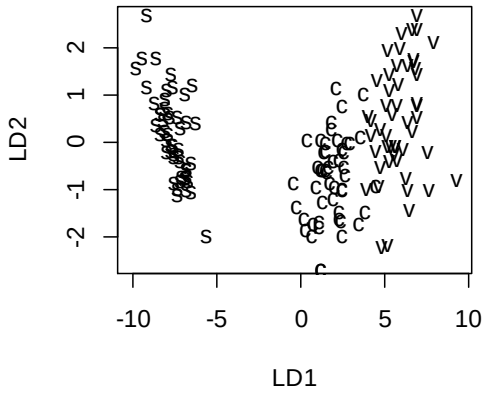
```
plot(lda.predict$x[,1], lda.predict$x[,2],
     col = lda.predict$class)

compare = data.frame(remote.sensing$crop,
                     lda.predict$class)
```



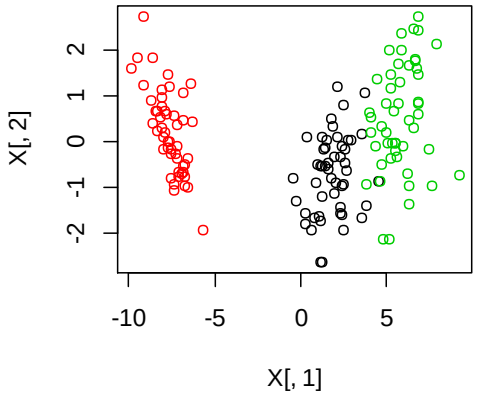
	remote.sensing.crop	lda.predict.class
1	CORN	CORN
2	CORN	CORN
3	CORN	CORN
4	CORN	CORN
5	CORN	CORN
6	CORN	SOYBEANS
7	CORN	CORN
8	SOYBEANS	SOYBEANS
9	SOYBEANS	SOYBEANS
10	SOYBEANS	SOYBEANS
11	SOYBEANS	SUGARBEETS
12	SOYBEANS	CORN
13	SOYBEANS	COTTON
14	COTTON	CLOVER
15	COTTON	SOYBEANS
16	COTTON	CLOVER
17	COTTON	SOYBEANS
18	COTTON	CLOVER
19	COTTON	COTTON
20	SUGARBEETS	CORN
21	SUGARBEETS	SOYBEANS
22	SUGARBEETS	SUGARBEETS
23	SUGARBEETS	CLOVER
24	SUGARBEETS	SOYBEANS
25	SUGARBEETS	SUGARBEETS
26	CLOVER	COTTON
27	CLOVER	SUGARBEETS
28	CLOVER	CLOVER
29	CLOVER	CLOVER
30	CLOVER	CLOVER
31	CLOVER	SUGARBEETS
32	CLOVER	CLOVER
33	CLOVER	CLOVER
34	CLOVER	COTTON
35	CLOVER	CLOVER
36	CLOVER	COTTON

R - linear discriminant analysis

```
library(MASS)
data(iris3)
Dat <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                  Sp = rep(c("s","c","v"), rep(50,3)))

lda <- lda(Sp ~ Sepal.L. + Sepal.W.+ Petal.L.+ Petal.W., Dat)
plot(lda, cex=1)
```




```
x <- lda; data <- model.frame(x)
Terms <- x$terms
X <- model.matrix(delete.response(Terms), data)
Y <- model.response(data)
xint <- match("(Intercept)", colnames(X), nomatch = 0L)
X <- X[, -xint, drop = FALSE] #remove intercept
means <- colMeans(x$means)
```




```
# linear discriminant factors
Score <- scale(X, center = means, scale = FALSE) %*% x$scaling #calculate LDs
plot(Score[,1], Score[,2], col=Dat$Sp)
```

Quadratic Discriminant Analysis

	Sepal L.	Sepal W.	Petal L.	Petal W.
[1,]	4.8	3.4	1.6	0.2
[2,]	5.8	4	1.2	0.2
[3,]	5	3.6	1.4	0.2
[4,]	4.4	2.9	1.4	0.2
[5,]	5.2	3.4	1.4	0.2
[6,]	5.1	3.5	1.4	0.2

```
library(MASS)
dim(iris3) # 50 4 3 (3 species)
tr <- sample(1:50, 25)
train <- rbind(iris3[ tr,,1], iris3[ tr,, 2], iris3[ tr,, 3])
test  <- rbind(iris3[-tr,,1], iris3[-tr,, 2], iris3[-tr,, 3])
cl <- factor(c(rep("s", 25), rep("c", 25), rep("v",2 5))) # 3 species

z <- qda(train, cl)
predict(z, test)$class
```

```
[1] s s s s s s s s s s s s s s s s s s s s s s s s s c c c c c c c c c c c c c c c c c c c c c c c v v v v v v
[57] v v v v v v v v v v c v v v v v v v v
Levels: c s v
```

Discriminant analysis vs. clustering

Discriminant Analysis	Clustering
• known number of classes	• unknown number of classes
• based on a training set	• no prior knowledge
• used to classify future observations	• used to understand (explore) data
• classification is a form of supervised learning	• clustering is a form of unsupervised learning
• Y = X1 + X2 + X3……	• X1 + X2 + X3……

Assignment

General objectives: learn about cluster analysis

- Develop a dataset to perform cluster analysis
- Describe your data, e.g. X1, X2, X3, etc.
- Plot and interpret the results.