# Unsupervised Learning

## Clustering

Do groups exist in the data?

Can we form groups or clusters where objects in a cluster are similar to each other and dissimilar to objects in other groups?

**Example 1**

Data on purchases of supermarket customers from loyalty cards. Form clusters who

- shop for families
- no kids
- healthy
- organic
- cost conscious

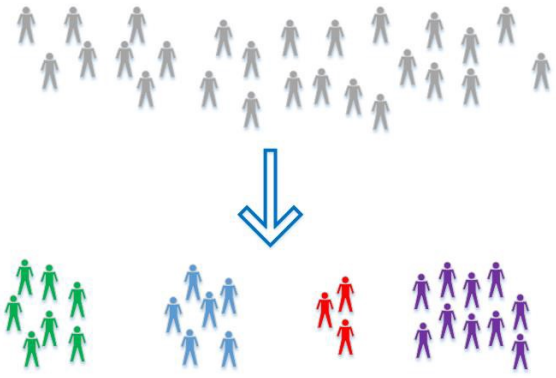
These are overlapping groups.

**Example 2**

Data on cDNA microarrays. Expression profiles for a specific gene sequence are recorded for $n$ individuals diagnosed with autism spectrum disorders. We could use clustering methods to form groups. Groups might represent subtypes of the disorder. These are disjoint clusters.

**Example 3**

Internet searching with Google. Organizes results into clusters. There is a "show similar pages" button.



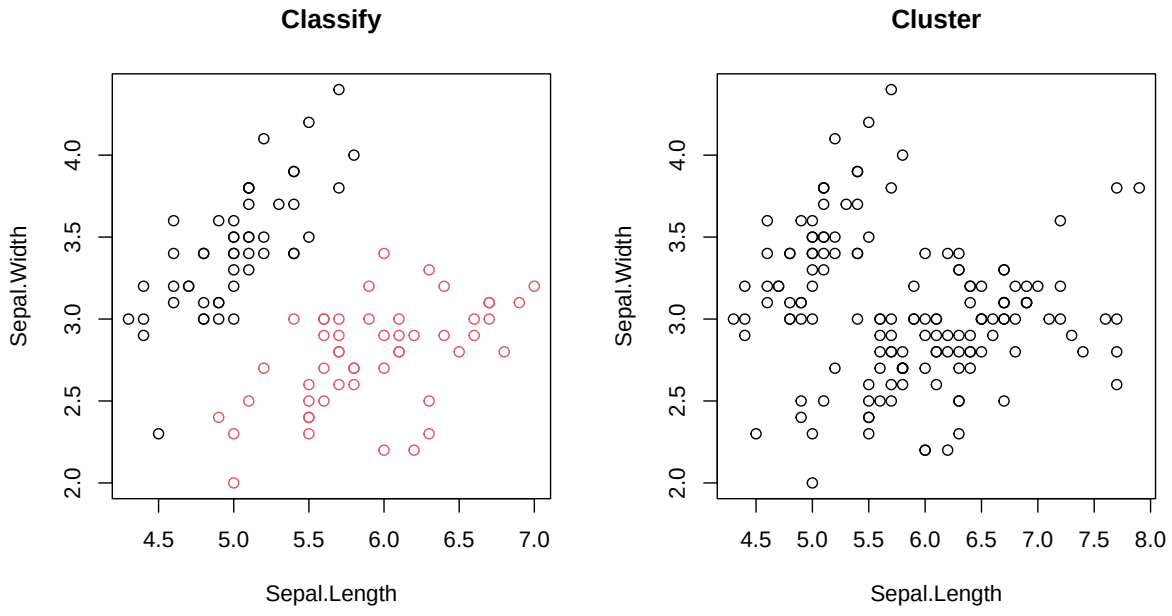
The classical clustering problem seeks to form a partition of the data where objects in a cluster are similar to each other and dissimilar to objects in other groups.

If we establish the presence of groups we are then interested in **how many** groups there are and what the make up of the groups is.

This is an example of **unsupervised learning**.

In **supervised learning**, you know the group membership (think of this as the response), and you want to relate this membership to other variables.

```
par(mfrow=c(1,2))
plot(iris[1:100,1:2], col=iris$Species[1:100], main="Classify")
plot(iris[,1:2],main="Cluster")
```




In the cluster analysis problem we have $n$ objects and we want to form $k$ clusters, for $1 \leq k \leq n$ where the clusters do not overlap. The clusters form a partition.

Ideally we would like to pick the best possible clustering by evaluating all possibilities and picking the best. This is computationally infeasible, unless $n$ is very small.
Instead, clustering algorithms are "greedy", in that they iterate towards a solution by taking the best step at each stage, but the result will not usually be the overall best.

## Similarity

How similar are two objects?
If the two objects are cases, we would consider them to be similar if the distance between them is small.
If the two objects are variables, we would consider them to be similar if the correlation or some other measure of association is high.

## Distance between cases

Consider case $i$ with coordinates $x_{i1}, x_{i2}, \ldots, x_{ip}$, and case $j$ with coordinates $x_{j1}, x_{j2}, \ldots, x_{jp}$.

1. The **Euclidean distance** between case $i$ and case $j$ is

$$\begin{aligned} d(i,j) &= \sqrt{(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots (x_{ip}-x_{jp})^2} \\ &= \sqrt{\sum_{k=1}^{p}(x_{ik}-x_{jk})^2} \end{aligned}$$

2. The **squared Euclidean distance** between case $i$ and case $j$ is

$$d(i,j) = \sum_{k=1}^{p}(x_{ik}-x_{jk})^2$$

3. The **Manhattan distance** between case $i$ and case $j$ is

$$d(i,j) = \sum_{k=1}^{p}|x_{ik}-x_{jk}|$$

**Note:**

- These measures are appropriate when variables are quantitative.
  There are other distance measures that are more appropriate when variables are categorical or binary.

- If the variables are measured in different units or have different scales then it is advisable to standardise variables prior to computing distances. ( Note: sometimes this may not be advisable! We may prefer to give less weight to a variable which carries less information (i.e. lower variance).
- The measures above are true distances in that they satisfy the properties
  1. $d(i,j) = d(j,i)$
  2. $d(i,j) \geq 0$ and $d(i,j) = 0$ iff case $i$ and $j$ are identical.
  3. $d(i,j) \leq d(i,k) + d(k,j)$ (the triangle inequality).
- Form a distance matrix. Distance matrices from the above are symmetric with zeros along the main diagonal.

A made up dataset

```
x <- matrix(c(3, 0, -1,
              5, 4, 0,
              3, 1, 2,
              3, 4, 1,
              2, 1, 4),nrow=5)
rownames(x)<- letters[1:5]
colnames(x)<- c("V1", "V2", "V3")
x
```

```
  V1 V2 V3
a  3  0  4
b  0  3  1
c -1  1  2
d  5  2  1
e  4  3  4
```

Euclidean distance

```
D1 <- as.matrix(dist(x, method = "euclidean"))
D1
```

```
    a   b   c   d   e
```

```
a 0.0 5.2 4.6 4.1 3.2
b 5.2 0.0 2.4 5.1 5.0
c 4.6 2.4 0.0 6.2 5.7
d 4.1 5.1 6.2 0.0 3.3
e 3.2 5.0 5.7 3.3 0.0
```

Squared Euclidean distance

```
D2 <- D1 * D1
D2
```

```
   a  b  c  d  e
a  0 27 21 17 10
b 27  0  6 26 25
c 21  6  0 38 33
d 17 26 38  0 11
e 10 25 33 11  0
```

Manhattan distance

```
D3 <- as.matrix(dist(x, method = "manhattan"))
D3
```

```
  a b c d e
a 0 9 7 7 4
b 9 0 4 6 7
c 7 4 0 8 9
d 7 6 8 0 5
e 4 7 9 5 0
```

## Similarity between variables

1. The **(Pearson) correlation** between variables $i$ and $j$:

$$r_{ij} = \frac{\sum_{k=1}^{n}(x_{ki} - \bar{x}_i)(x_{kj} - \bar{x}_j)}{(n-1)s_i s_j}$$

$$= \frac{\sum_{k=1}^{n} x_{ki} x_{kj}}{n-1}, \text{ if the variables are standardised to mean 0, sd 1}$$

2. The **Spearman or rank correlation** between variables $i$ and $j$: First rank the variables.

```
rx <- apply(x, 2, rank)
rx
```

```
  V1  V2  V3
a  3 1.0 4.5
b  2 4.5 1.5
c  1 2.0 3.0
d  5 3.0 1.5
e  4 4.5 4.5
```

Then compute the Pearson correlation on the ranks.

```
cor(rx)
```

```
     V1    V2    V3
V1 1.00  0.21  0.00
V2 0.21  1.00 -0.32
V3 0.00 -0.32  1.00
```

Actually, Spearman correlation is more easily calculated as

```
cor(x, method = "spearman")
```

```
     V1    V2    V3
V1 1.00  0.21  0.00
V2 0.21  1.00 -0.32
V3 0.00 -0.32  1.00
```

3. Sometimes similarity between variables is measured by absolute correlation.

To transform a similarity matrix to a dissimilarity matrix use $D = c - S$, for some constant $c$.

## Clustering algorithms

Algorithms are of the following types:

1. Hierarchical clustering.

   This can be agglomerative (bottom up) or divisive (top down)

2. Partitioning methods

   Divide objects into $k$ clusters. Re-allocate objects until assignment does not improve there exult.

3. Optimization

   Search for optimal value of some goodness of clustering criterion. Partitioning methods are a special case.

## Hierarchical clustering (agglomerative)

You have $n$ objects. You wish to form $k$ clusters.
Construct a distance matrix $D$.

1. Start with $m = n$ clusters.

2. Find the closest pair of clusters and merge them. Now there are $m = n - 1$ clusters.

3. Repeat step 2 until $m = k$

Usually, you use $k = 1$. Then, examine the sequence of partitions constructed and pick "the best" one.
Step 2 above requires a distance measure between two clusters.

Suppose we have two clusters $U = u_1, u_2, \ldots u_r$ and $V = v_1, v_2, \ldots v_s$.

**Single linkage distance:** Define

$$d(U, V) = \min_{u \in U, v \in V} d(u, v)$$

This is the smallest distance between a point in $U$ and a point in $V$.

Consider the distance

Consider

```
D <- D2

D
```

```
   a  b  c  d  e
a  0 27 21 17 10
b 27  0  6 26 25
c 21  6  0 38 33
d 17 26 38  0 11
e 10 25 33 11  0
```

$U = \{a, b, c\}$, $V = \{d, e\}$

What is the single linkage distance between $U$ and $V$?

**Complete linkage distance:** Define

$$d(U, V) = \max_{u \in U, v \in V} d(u, v)$$

This is the largest distance between a point in $U$ and a point in $V$.

$U = \{a, b, c\}$, $V = \{e, f\}$

What is the complete linkage distance between $U$ and $V$?

**Average linkage distance:** Define

$$d(U, V) = \frac{\sum_{u \in U, v \in V} d(u, v)}{rs}$$

This is the average distance between points in $U$ and points in $V$.

**Distance between centroids:** Let $\mathbf{u}$ be the centroid of cluster U and $\mathbf{v}$ be the centroid of cluster V. Define

$$d(U, V) = d(\mathbf{u}, \mathbf{v})$$

where $d$ is a distance measure.

The centroids are

```
U <- c("a","b","c")
V <- c("d","e")
u <- colMeans(x[U,])
u
```

```
  V1   V2   V3
0.67 1.33 2.33
```

```
v <- colMeans(x[V,])
v
```

```
 V1  V2  V3
4.5 2.5 2.5
```

The distance between the centroids is

```
dist(rbind(u, v))
```

```
  u
v 4
```

Agglomerative hierarchical clustering using the single linkage distance is called single linkage clustering.
Agglomerative hierarchical clustering using the complete linkage distance is called complete linkage clustering, etc.

### Single linkage clustering

```
D
```

```
   a  b  c  d  e
a  0 27 21 17 10
b 27  0  6 26 25
c 21  6  0 38 33
d 17 26 38  0 11
e 10 25 33 11  0
```

1. Find closest pair of objects.

   They are c and b at a distance of 6

   Place these two in the same cluster.

   Clusters are now a, bc, d,e.

2. Re calculate distance between clusters.

```
D


    a bc  d  e
a   0 21 17 10
bc 21  0 26 25
d  17 26  0 11
e  10 25 11  0
```

3. Find closest pair of objects.

   They are e and a at a distance of 10

   Place these two in the same cluster.

   Clusters are now ae, bc, d.

4. Re calculate distance between clusters.

```
D


   ae bc  d
ae  0 21 11
bc 21  0 26
d  11 26  0
```

5. Find closest pair of objects.

   They are d and ae at a distance of 11

   Place these two in the same cluster.

   Clusters are now aed, bc.

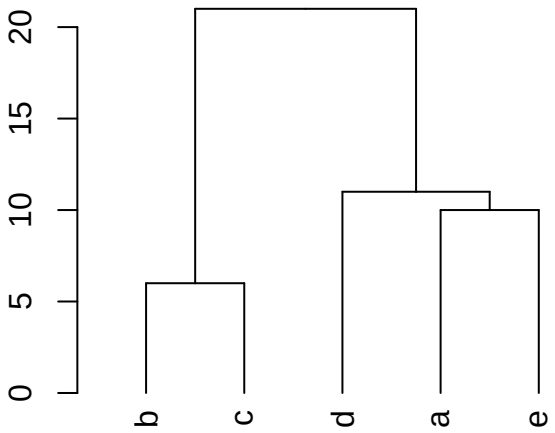
6. Re calculate distance between clusters.

```
D


    aed bc
aed   0 21
bc   21  0
```

7. Merge final two clusters, distance is 21

Single linkage in R is calculated as

```
h<- hclust(as.dist(D2),"single")
h



Call:
hclust(d = as.dist(D2), method = "single")

Cluster method   : single
Number of objects: 5

plot(as.dendrogram(h))
```




The above plot shows the series of joins in the clustering process.

The x-axis shows the objects being clustered.

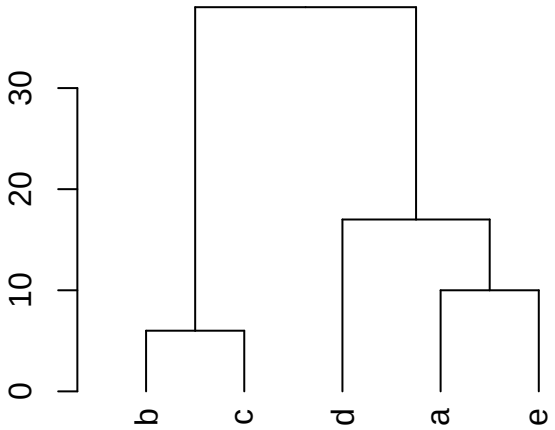
The y-axis shows the distance at which objects or clusters are joined.

The objects along the x-axis are ordered in such a way that there is no crossing, i.e. joined objects or clusters are placed adjacently.

The plot is called a *dendrogram*.

The results of complete linkage in R are

```
h<- hclust(as.dist(D2),"complete")
d <- as.dendrogram(h)
plot(d)
```




## Example – Eurostat

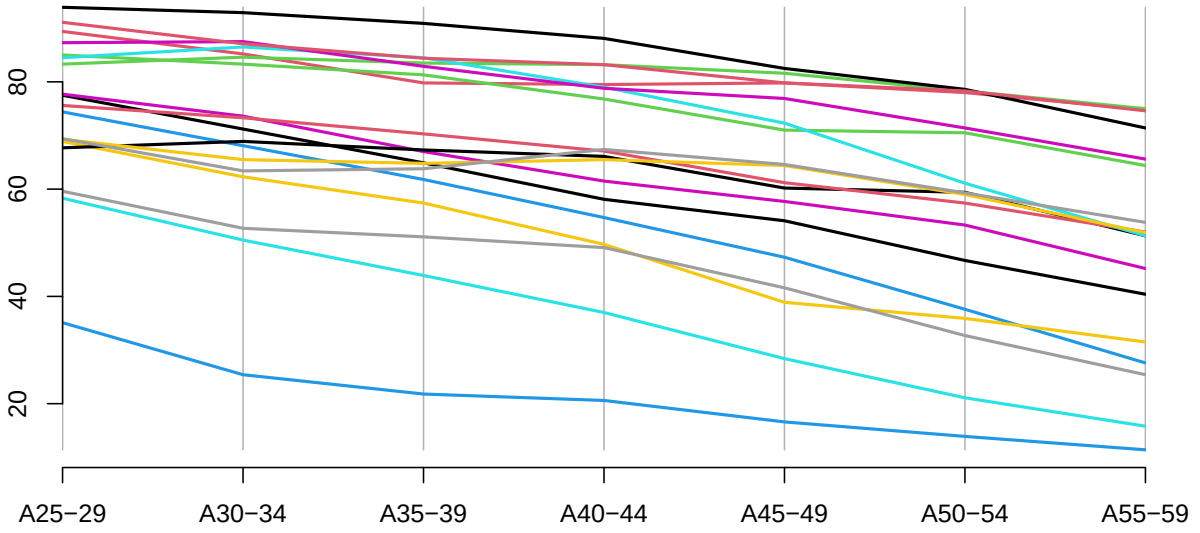
In 1999, Eurostat compiled data on the percentage of the population in each age group in the EU-15 and 3 other countries who had completed at least the upper secondary education.

```
educ <- read.table("https://raw.githubusercontent.com/rafamoral/courses/main/intro_sta
                   row.names = 1, header = TRUE)
educ <- educ[,-19]
educ <- t(educ)
head(educ)
```

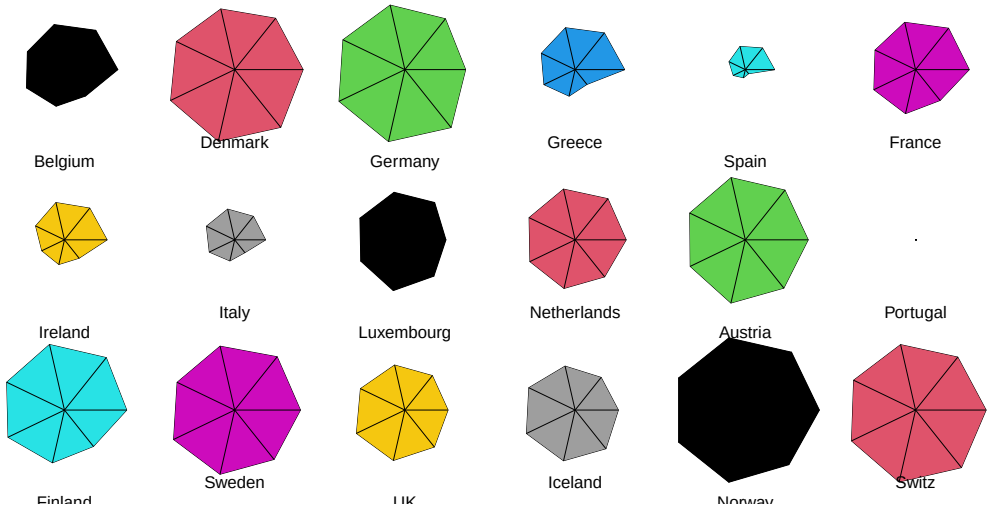
```
        A25-29 A30-34 A35-39 A40-44 A45-49 A50-54 A55-59
Belgium     78     71     65     58     54     47     40
Denmark     89     85     80     80     80     78     75
Germany     83     85     84     83     82     78     75
Greece      74     68     62     55     47     38     28
```

```
Spain       58      50      44      37      28      21      16
France      78      74      67      62      58      53      45
```

```
library(PairViz)
pcp(educ,lwd=2, scale=FALSE, col=1:18)
axis(2)
```



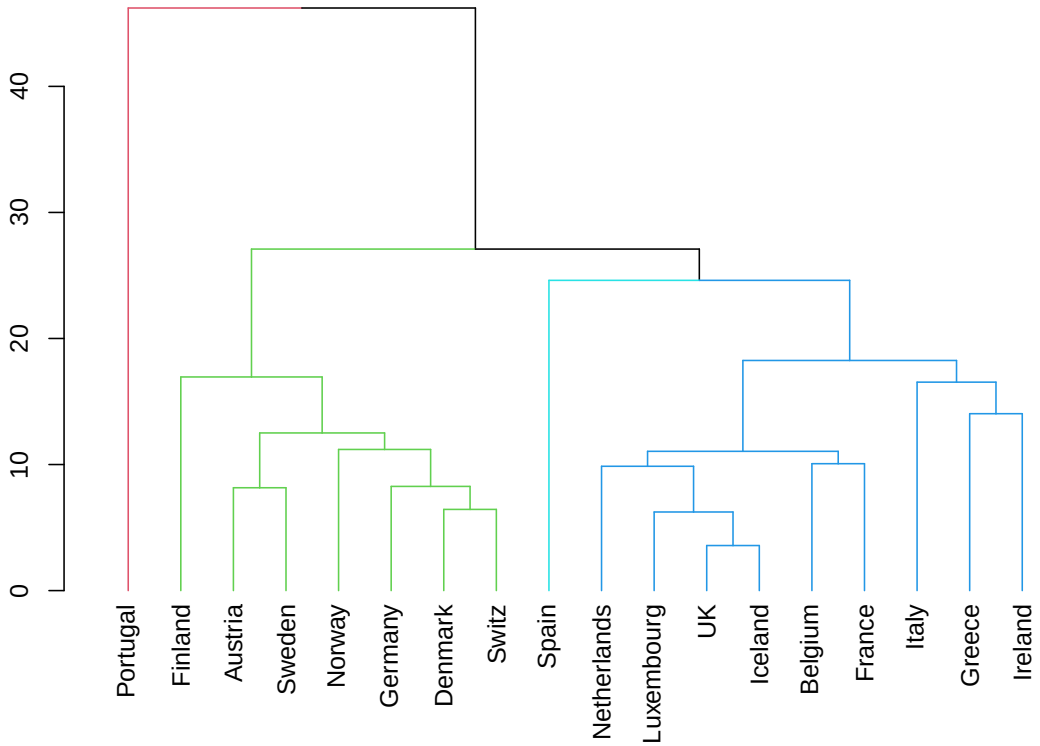

```
stars(educ,nrow=3,col.stars=1:18)
```




There is clearly a big difference amoung the countries.

**Single linkage clustering, no standardisation**

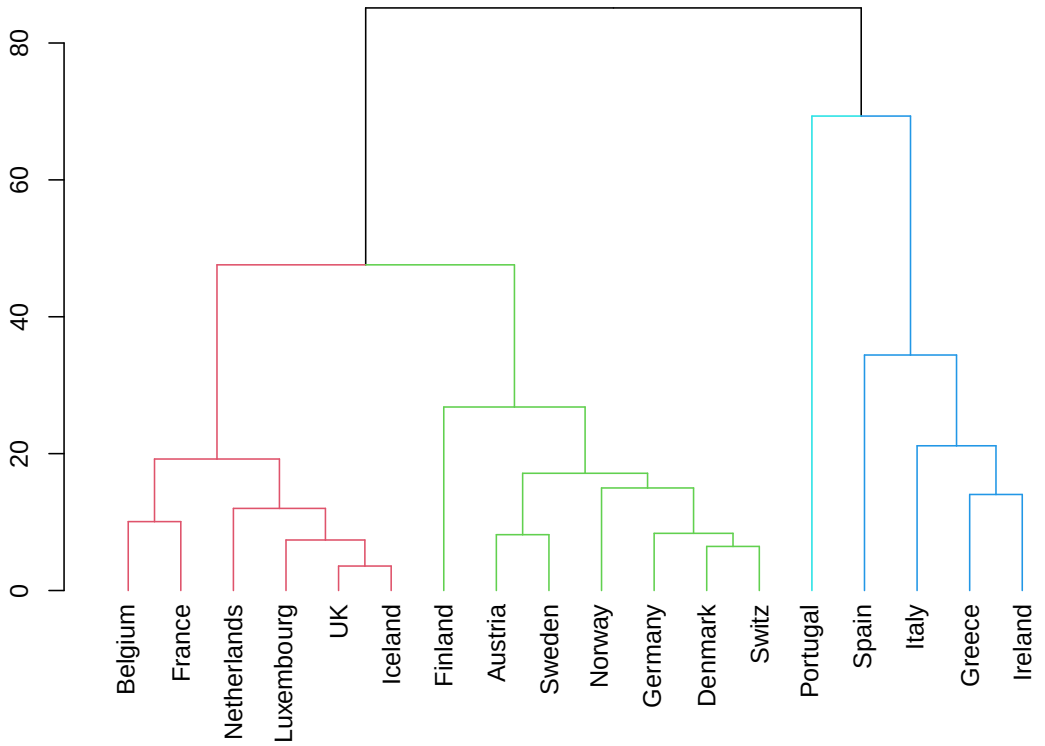
```
d <- dist(educ, "euclidean")
h <- hclust(d, "single")

d1 <- as.dendrogram(h)
# plot(d1)

# for a fancier plot with colors:
library(dendextend)

d2 <- color_branches(d1, k = 4, col = c(2,3,5,4)) # auto-coloring 4 clusters of brano
plot(d2)
```




If we use average linkage, then the clusters are somewhat different:

```
d <- dist(educ, "euclidean")
h <- hclust(d, "average")
d1 <- as.dendrogram(h)

d2 <- color_branches(d1,k=4,col=c(2,3,5,4)) # auto-coloring 4 clusters of branches.
```

```
plot(d2)
```



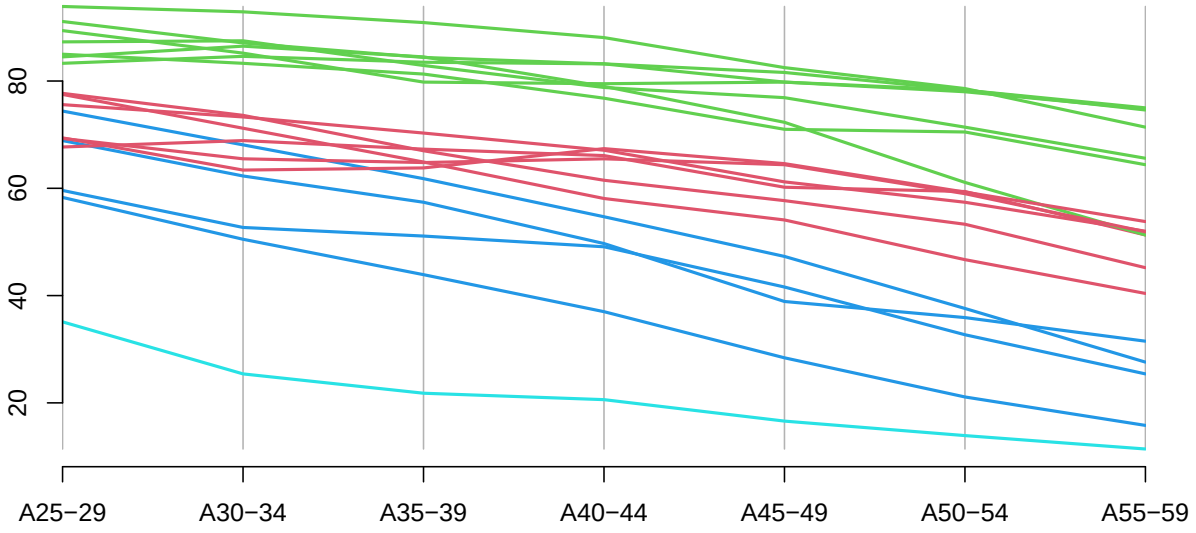

Using 4 clusters find the cluster membership

```
cutree(h, 4)
```

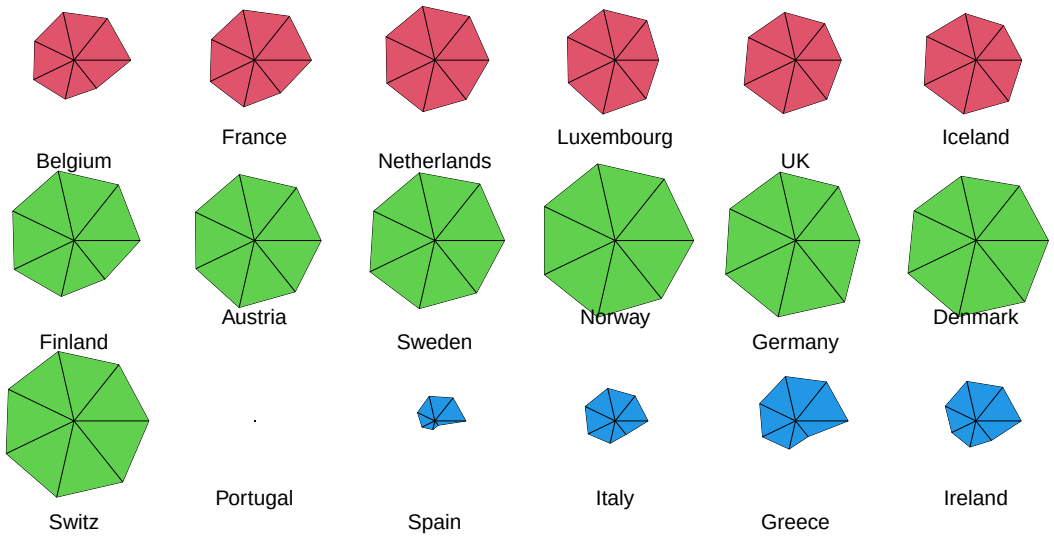
```
    Belgium     Denmark     Germany      Greece       Spain      France
          1           2           2           3           3           1
    Ireland       Italy  Luxembourg Netherlands     Austria    Portugal
          3           3           1           1           2           4
    Finland      Sweden          UK     Iceland      Norway      Switz
          2           2           1           1           2           2
```

We can explore clusters graphically and also numerically.

```
pcp(educ, lwd = 2, scale = FALSE, col = cutree(h,4) + 1)
axis(2)
```




```
stars(educ[h$order,], nrow = 3, col.stars = cutree(h,4)[h$order] + 1)
```




Portugal is in a cluster of its own with the lowest education level in all age groups

The next group is the blue group.

The red group are next, followed by green which have the highest attainment.

## Numerical summaries of clusters

Ideally all objects in a cluster should be close to each other and far away from objects in other clusters.

1. How close are objects in a cluster? Let $d$ represent Euclidean distance. Measure by

(a) Maximum inter-object distance. For cluster $C_i$ this is

$$\max_{\mathbf{x_j},\mathbf{x_k}\in\mathbf{C_i}} d(\mathbf{x_j},\mathbf{x_k})$$

(b) Average inter-object distance. For cluster $C_i$ this is

$$\text{avg}_{\mathbf{x_j},\mathbf{x_k}\in\mathbf{C_i}} d(\mathbf{x_j},\mathbf{x_k})$$

(c) Average distance from centroid

Let:

- $\mathbf{m_i}$ be the **centroid** of cluster $i$ and
- $d(\mathbf{x},\mathbf{m_i})$ denote the **Euclidean distance** from $\mathbf{x}$ to $\mathbf{m_i}$.

The average distance to centroid for cluster $C_i$ which has $n_i$ objects is

$$\sum_{\mathbf{x}\in\mathbf{C_i}} d(\mathbf{x},\mathbf{m_i})/\mathbf{n_i}$$

(d) Maximum distance from centroid

The maximum distance to centroid for cluster $i$ $C_i$ is

$$\max_{\mathbf{x}\in\mathbf{C_i}} d(\mathbf{x},\mathbf{m_i})$$

(e) Within cluster sum of squares, i.e.

$$\sum_{\mathbf{x}\in\mathbf{C_i}} d(\mathbf{x},\mathbf{m_i})^{\mathbf{2}}$$

2. How far apart are two clusters?

(a) Any of the distances single linkage, complete linkage, average linkage etc presented earlier could be used.

(b) Use the distance between cluster centroids.

$$d(\mathbf{m_i},\mathbf{m_j})$$

```
sumPartition(educ,cutree(h,4))

Final Partition

Number of clusters  4

          N.obs Within.clus.SS Ave.dist..Centroid Max.dist.centroid
Cluster 1     6            630                9.8                16
Cluster 2     7           1139               11.7                22
Cluster 3     4           1220               16.1                25
Cluster 4     1              0                0.0                 0



Cluster centroids

       Cluster 1 Cluster 2 Cluster 3 Cluster 4 Grand centrd
A25-29        73        88        65        35           75
A30-34        69        87        58        25           71
A35-39        66        84        54        22           68
A40-44        64        81        48        21           65
A45-49        60        78        39        17           60
A50-54        56        74        32        14           55
A55-59        49        68        25        11           49



Distances between Cluster centroids

          Cluster 1 Cluster 2 Cluster 3 Cluster 4
Cluster 1         0        46        47       111
Cluster 2        46         0        92       157
Cluster 3        47        92         0        69
Cluster 4       111       157        69         0
```

In this example, interpretation is straightforward.

1. Which cluster has education levels above the centroid? Below the centroid?

2. Which cluster is the most spread out? least spread out?

3. Which pair of clusters are closest? farthest away?

## Comments

- Single and complete linkage will give unchanged results when distances are transformed by a monotone transformation.
  This invariance property does not hold when group average linkage is used.

- Single linkage cannot detect poorly separated clusters.

- Single linkage often gives long straggly clusters where objects at distant ends have low similarity. This is knopwn as chaining.

- Average linkage clustering is generally recommended.

- Algorithms are fast: time proportional to square of number of objects: $O(n^2)$.

## Example – Music data

**Number of cases:** 62

**Number of variables:** 7

**Description:** Music tracks were read into music editing software and the first 40 second clip was converted into numeric data. All of the CDs contained left and right channels, and variables were calculated on the left channel.

| Variable | Explanation |
|---|---|
| artist | Abba, Beatles, Eels, Vivaldi, Mozart, Beethoven, Enya |
| type | rock, classical or new wave. |
| lvar, lave, lmax | variance, average, maximum of the frequencies of the left channel |
| lfener | an indicator of the amplitude or loudness of the sound |
| lfreq | median location of the 15 highest peaks in the periodogram. |

**Questions:**

- Can we distinguish between rock and classical tracks?

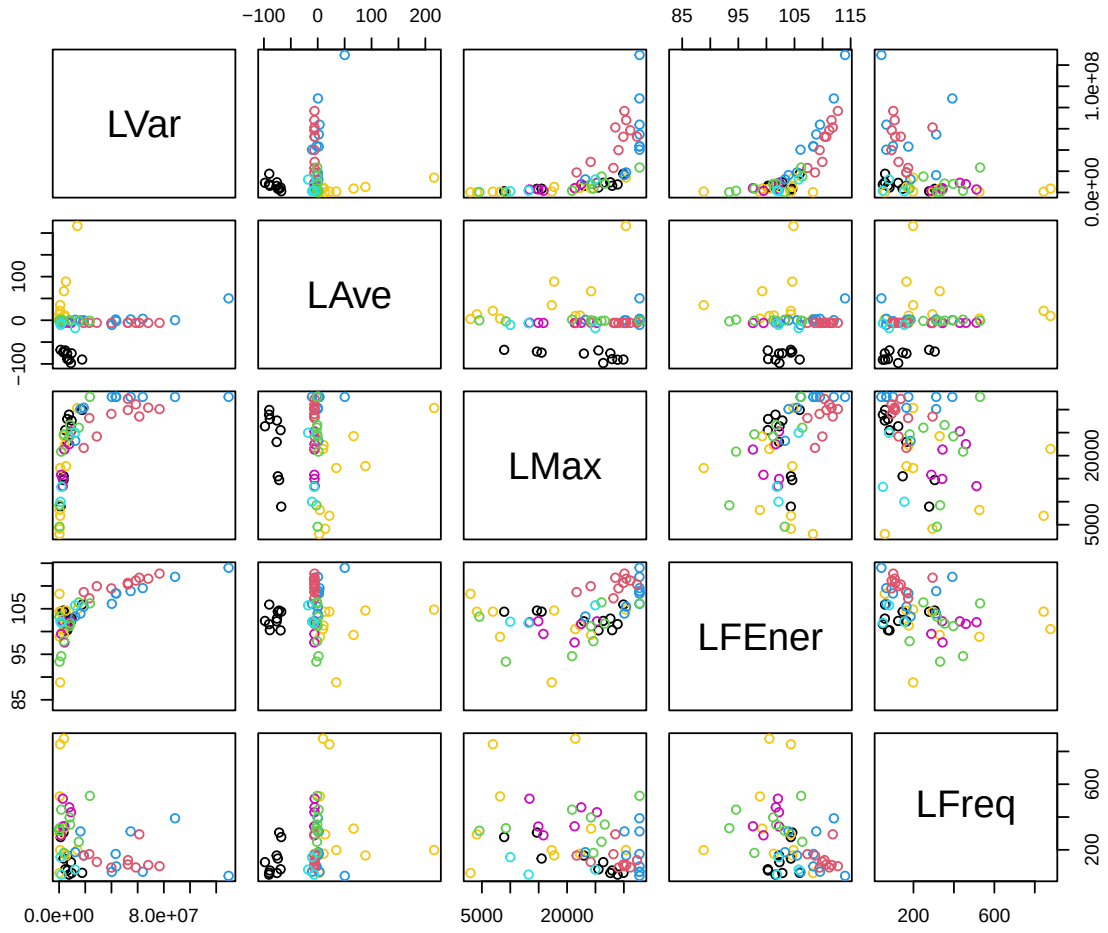
- Can we group the tracks into a small number of clusters according to their similarity on audio characteristics?

- Are there differences between the tracks of different artists?
- Answers might be used to arrange tracks on a digital music player, or to make recommendations.

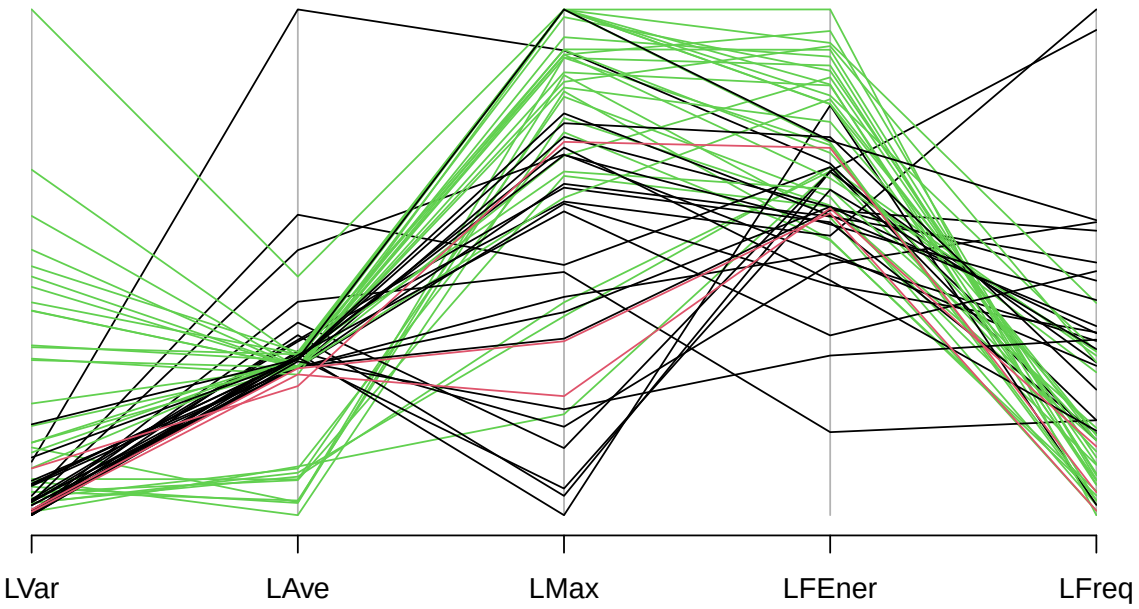
| Song | Artist | Type | LVar | LAve | LMax | LFEner | LFreq |
|---|---|---|---|---|---|---|---|
| Dancing Queen | Abba | Rock | 17600755.6 | -90.01 | 29921 | 105.92 | 59.57 |
| Knowing Me | Abba | Rock | 9543020.9 | -75.77 | 27626 | 102.84 | 58.48 |
| Take a Chance | Abba | Rock | 9049481.5 | 98.06 | 26372 | 102.32 | 124.59 |
| Mamma Mia | Abba | Rock | 7557437.3 | -90.47 | 28898 | 101.62 | 48.77 |

Table 1: Snapshot of 4 cases of the 'Music' multivariate data.

```
music <- read.csv("https://raw.githubusercontent.com/rafamoral/courses/main/intro_stat
music.feat <- music[, 4:8]
pairs(music.feat, col = music$Artist)
```

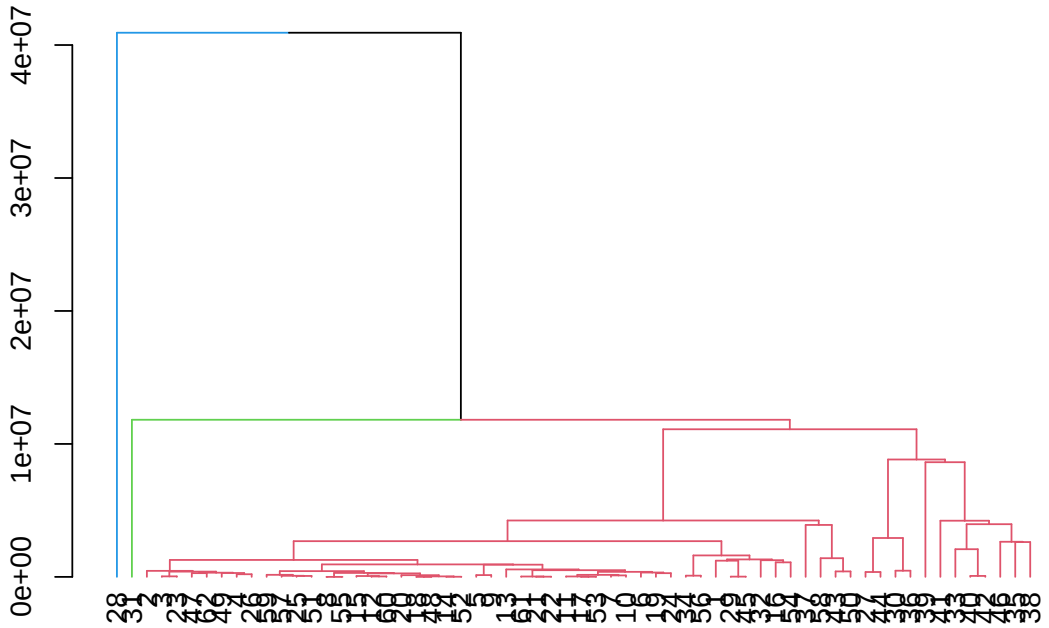


```
pcp(music.feat, col = music$Type)
```



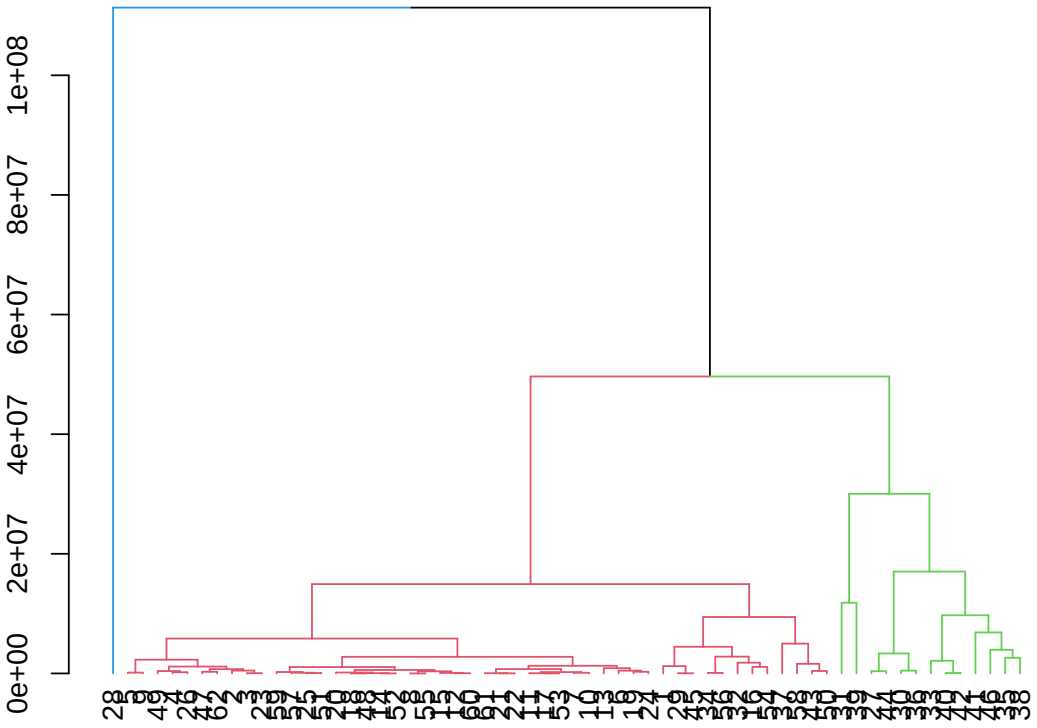

First use single linkage:

```
d <- dist(music.feat, "euclidean")
h <- hclust(d, "single")
labl.single <- cutree(h,3)
d1 <- as.dendrogram(h)

par(mar = c(10,4,3,2))
d2 <- color_branches(d1, k = 3, col = c(4,3,2)) # auto-coloring 4 clusters of branche
plot(d2)
```



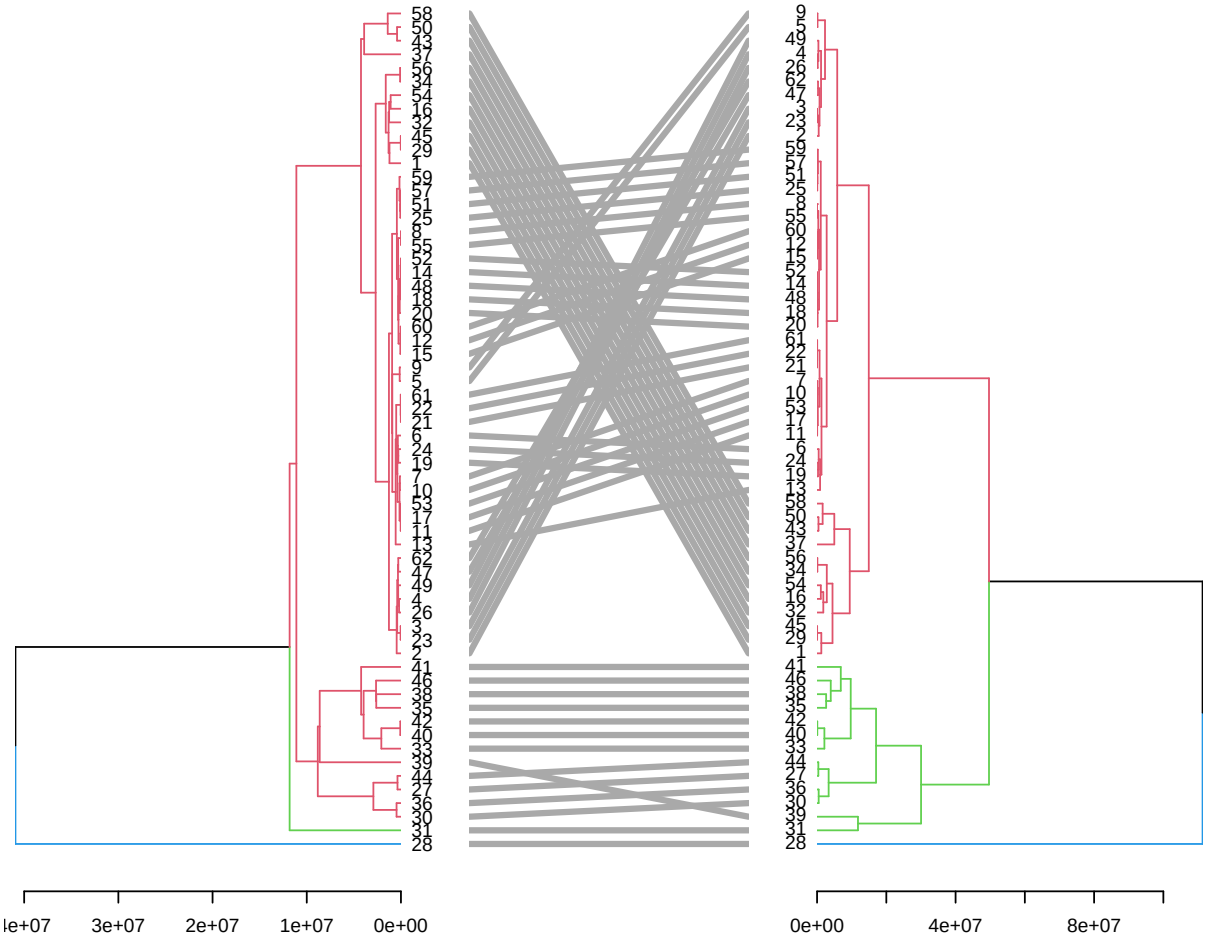

This exhibits chaining. Try average linkage.

```
h <- hclust(d, "average")
labl.Ave <- cutree(h, 3)
d3 <- as.dendrogram(h)

d4 <- color_branches(d3,k=3,col=c(4,2,3)) # auto-coloring 4 clusters of branches.
plot(d4)
```



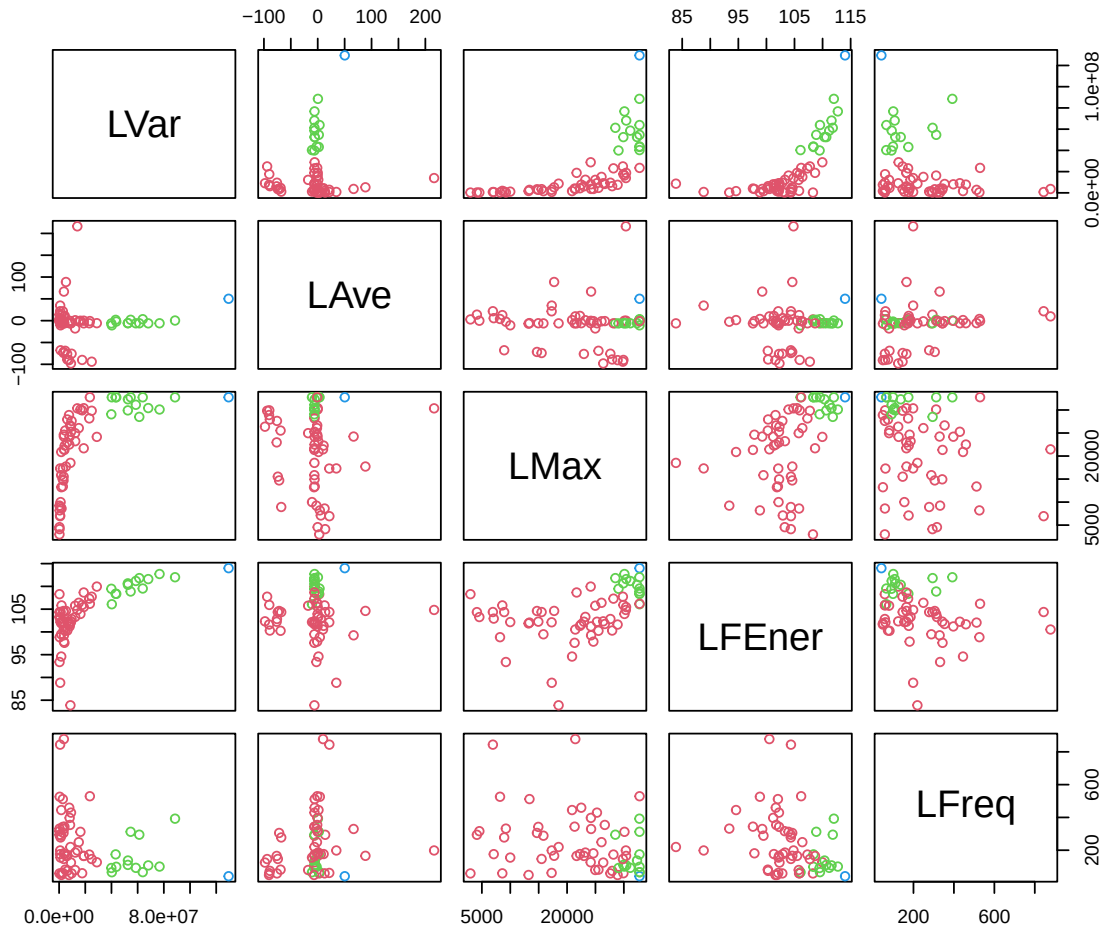

Average linkage produces a better clustering. Compare the clusters:

```
dl <- dendlist(d2, d4)



tanglegram(dl, sort = TRUE, common_subtrees_color_lines = FALSE,
           highlight_distinct_edges  = FALSE, highlight_branches_lwd = FALSE)
```

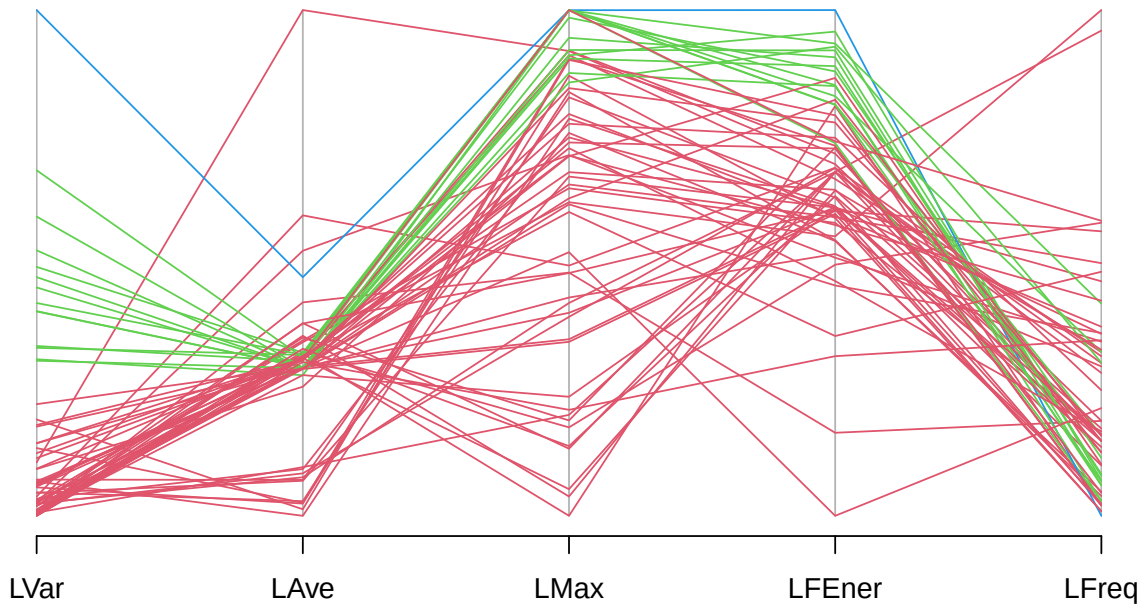



Colour the data by clusters obtained by average linkage clustering.

```
pairs(music.feat, col = labl.Ave+1)
```




```
pcp(music.feat, col = labl.Ave+1)
```

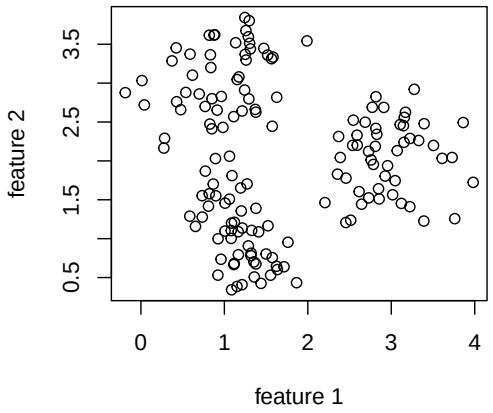



First variable (LVar) dominates the clustering.

# K-means clustering

K-means is a simple algorithm for partitioning a data set into K non-overlapping clusters.

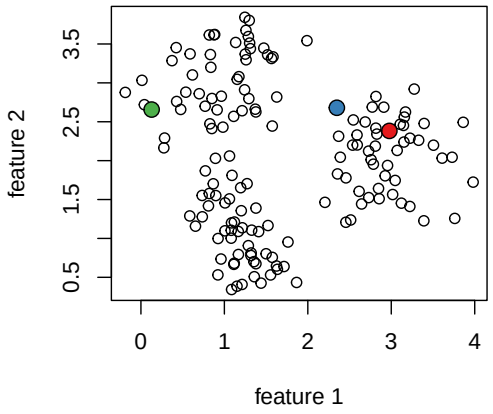
Consider the following simulated data.




We want to cluster the data into, say, three groups.

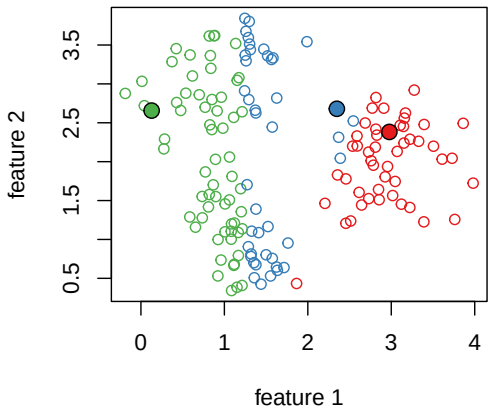
**$k$-Means Clustering: Iteration 0a**

We start by randomly generating three centers.



### $k$-Means Clustering: Iteration 0b

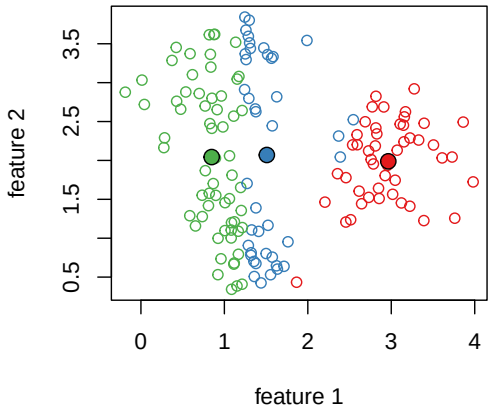
We label points according to which center is closest.




Closest center = smallest squared Euclidean distance

### $k$-Means Clustering: Iteration 1a

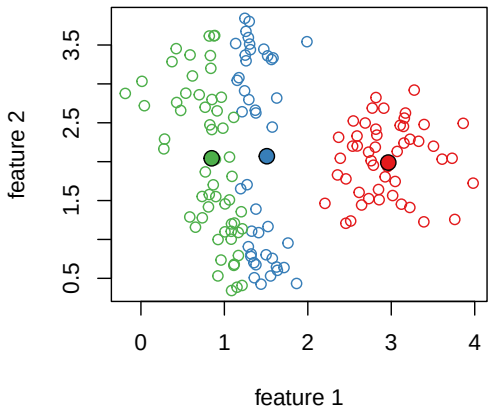
We update the values for the three centers.




Update = new centres for the groups are the mean of the values assigned to the group.

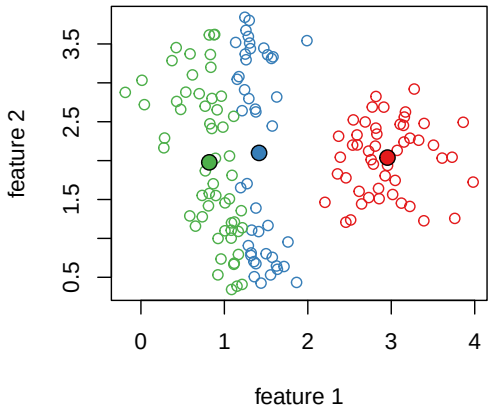
### $k$-Means Clustering: Iteration 1b

We label points according to which center is closest.



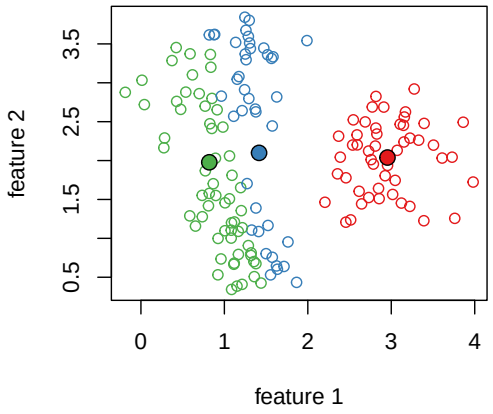

### $k$-Means Clustering: Iteration 2a

We update the values for the three centers.




### $k$-Means Clustering: Iteration 2b

We label points according to which center is closest.



**Convergence**

The $k$-Means algorithm converges when no points are moved between groups on an iteration. Once this happens, the estimates of the centers will no longer change, nor will the allocation of points to groups thereafter.

What I have just described is the default version of the algorithm implemented in R's kmeans() function.

Many variations exist, e.g. Lloyd's, (in R choose algorithm = "Lloyd" or "Forgy")

In this version, in the first step points are assigned to one of three clusters at random.

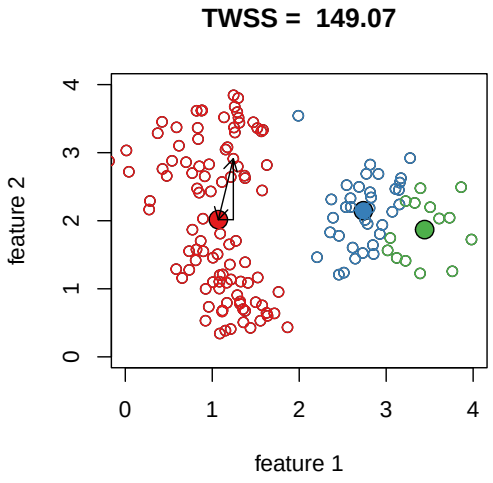
**Concepts**

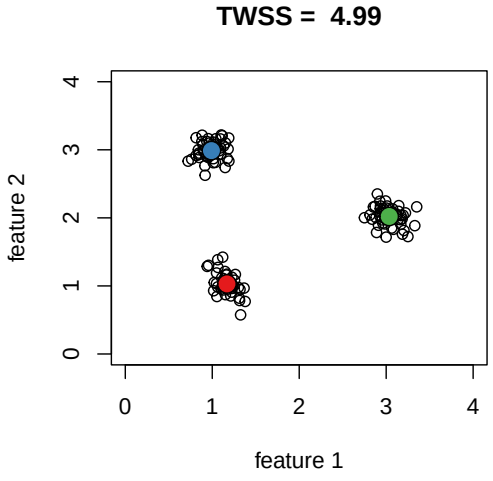
What is a **center** of a cluster? The mean of the values assigned to that cluster.

What do we mean by **closest**? Smallest squared Euclidean distance.

How well did we do? Compact clusters will have observations close to the center, sum of squared distances to the center will be small.

**TWSS**: sum of squared distances to the cluster center for all observations.



K-means attempts to find the assignment of objects to clusters that minimises the total within cluster variation, as measured by

$$\sum_{i=1}^{k} W(C_i)$$

where

$$W(C_i) = \sum_{x \in C_i} d(x, m_i)^2$$

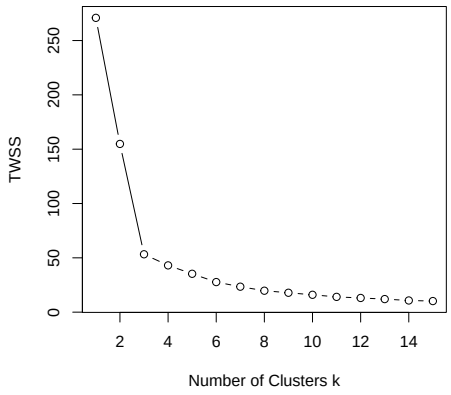
and $m_i$ is the centroid of cluster $i$.

That is, an observation $\mathbf{x}_i$ is assigned to group $k$ provided that $d(\mathbf{x}_i, \mu_k)$ is minimized,

$$d(\mathbf{x}_i, \mu_k) = \sum_{j=1}^{p} (x_{ij} - \mu_{kj})^2.$$

**How many clusters?**

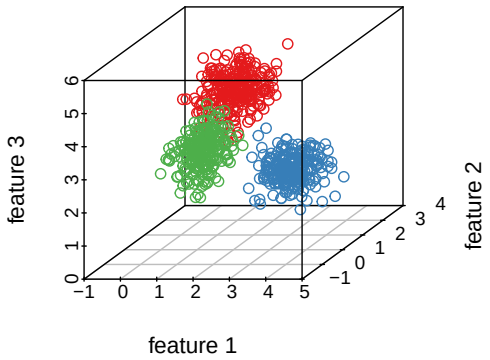
If we plot the total of the within sum of squares values versus $k$, then we get the following:




Notice that the graph flattens very quickly. What $k$ would you use?

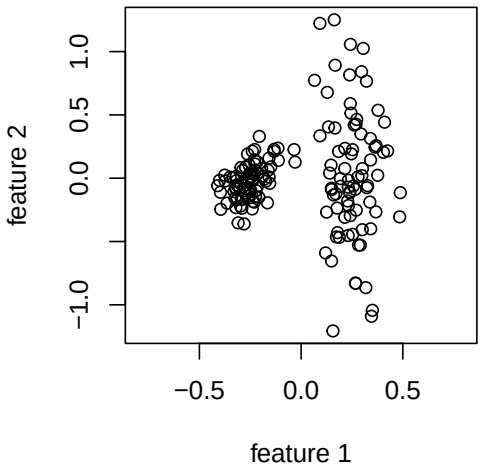
$p > 2$

It works for higher dimensional datasets.



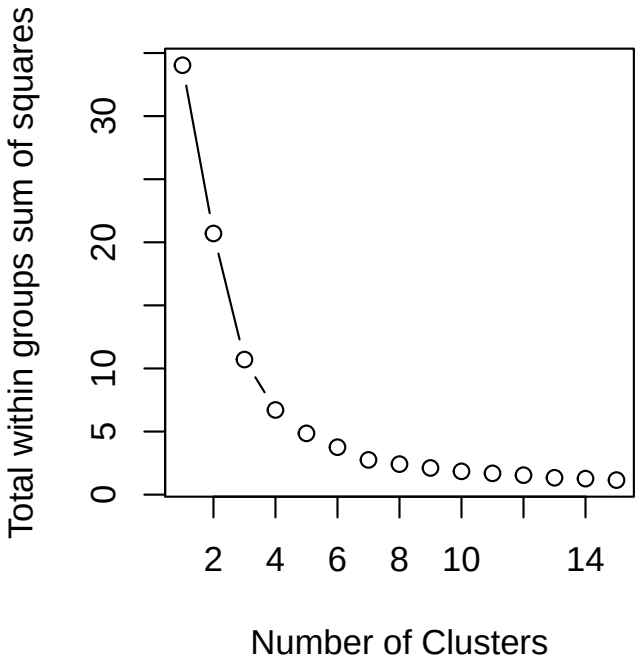
**Elliptical clusters**

If we run $k$-means on the following data.



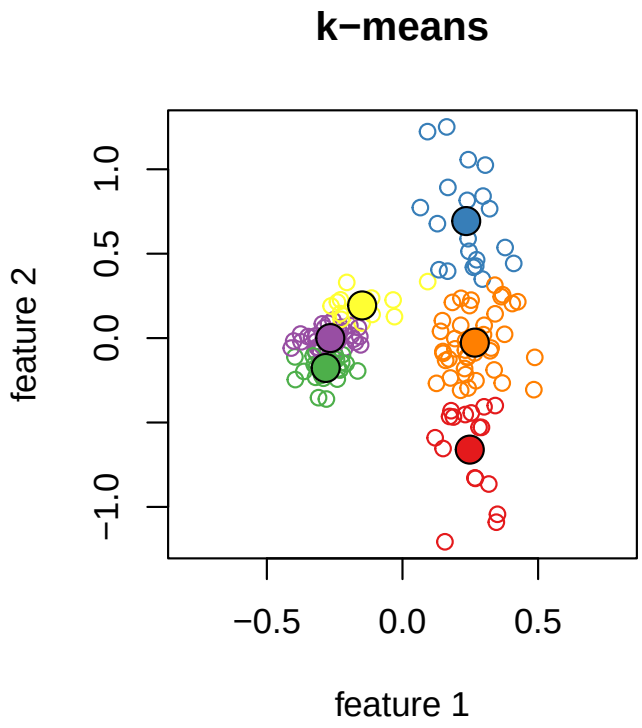

It looks like there are two groups.

If we plot the within sum of squares values versus $k$.




What value of $k$ looks good?

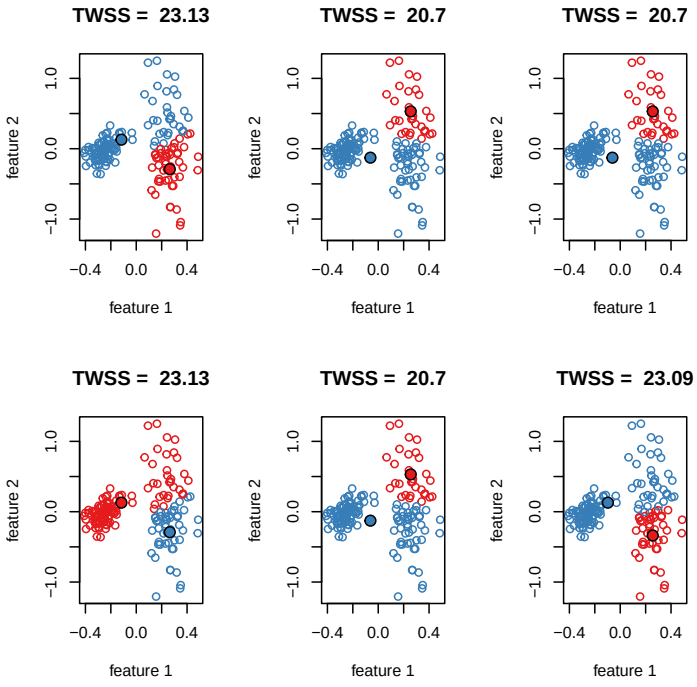
$K = 6$ gives the following clustering of the data




The elliptical groups are broken into subgroups. $k$-means clustering looks for circular clusters.

**Local minima**

The $k$-means algorithm may get give different answers when started at different values.




This means that it doesn't always find the minimum value for the Total Within Sum of Squares. Try a number of random starting partitions and pick the best (kmeans() in R does this if you specify nstart).

*Example: NCI60 data*