Clustering

LZ

May 7, 2019

- Meaningfull and usefull
- The purpose of clustering (summarization, efficiently finding Nearest Neighvors)
- It is **not** easy to cluster your data.

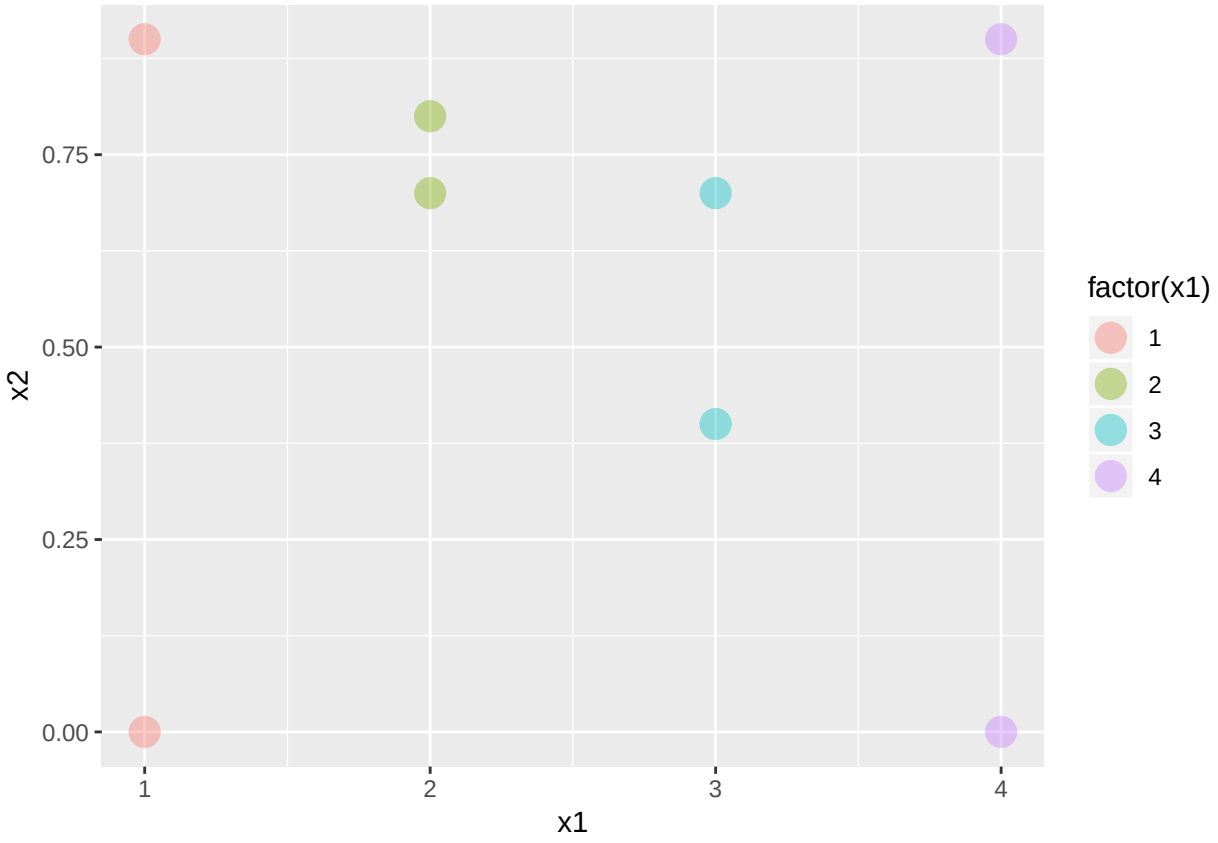
Different Types of Clusterings

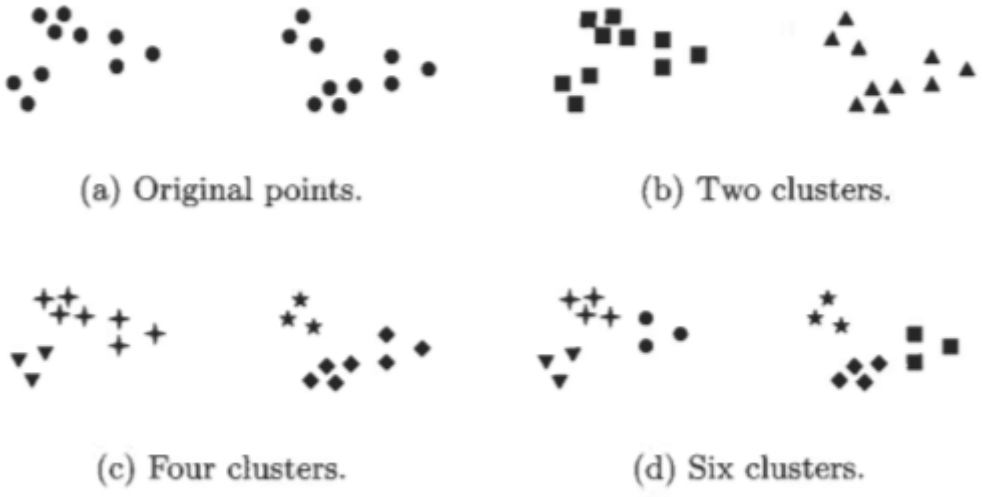
1) *Exclusive* versus *Overlapping* versus *Fuzzy*

- *Exclusive* - assign each object to a single cluster
- Point could reasonably be placed in more than one cluster (non-exclusive clustering). Simultaneously belong to more than one group. (2)
- Every object belongs to every cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belongs). - does not address true multiclass situations.

F1 = {(1, 0.9), (2,0.8), (3, 0.7), (4,0)} F2 = {(1,0), (2, 0.7), (3, 0.4), (4,0.9)}

```
x1 <- c(1,2,3,4,1,2,3,4)
x2 <- c(0.9, 0.8, 0.7, 0, 0, 0.7, 0.4, 0.9)
library(ggplot2)
ggplot(data.frame(x1,x2), aes(x1,x2, col = factor(x1)))+
  geom_point(size = 5, alpha = 0.4)
```





(a) Original points. (b) Two clusters.

(c) Four clusters. (d) Six clusters.

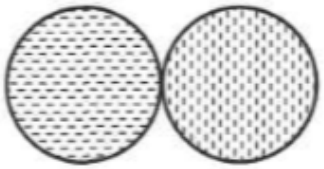
Figure 1:

2) *Complete* versus *partial*

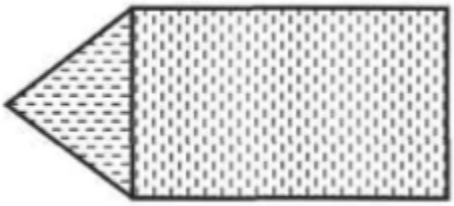
- A *complete* clustering assigns every object to a cluster
- The motivation for a partial clustering is that some objects in a data set may not belong to well-defined groups. (noise, outliers)

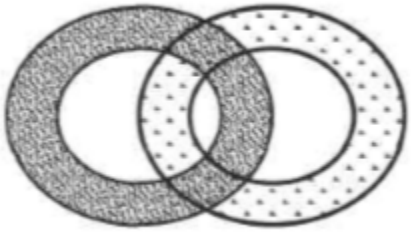
Different Types of Clusters

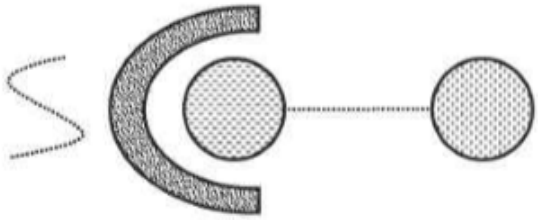
(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

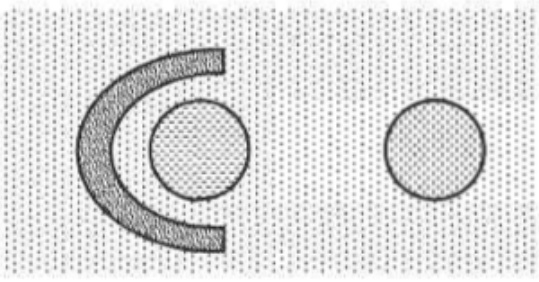


(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.









- While a **centroid** *almost never* corresponds to an **actual** data point, a **medoid**, by its definition, must be an **actual data** point.
- Intra-cluster distances are minimized (homogeneuos)
- Inter-cluster distances are maximized (heterogeneous)

Methods

- Hierarchical clustering
  - Agglomerative Methods
  - Divisive Methods
- Partitioning based clustering

Given k, the k-means algorithm consists of four steps:

- Select initial centroids at random.
- Assign each object to the cluster with the nearest centroid.
- Compute each centroid as the mean of the objects assigned to it.
- Repeat previous 2 steps until no change.

The idea behind K-means clustering is that a good clustering is one for which the **within-cluster variation** is as small as possible.

With this steps K-means algorithm maximizes Between group Sum of Squares and minimizes within group Sum of Squares.

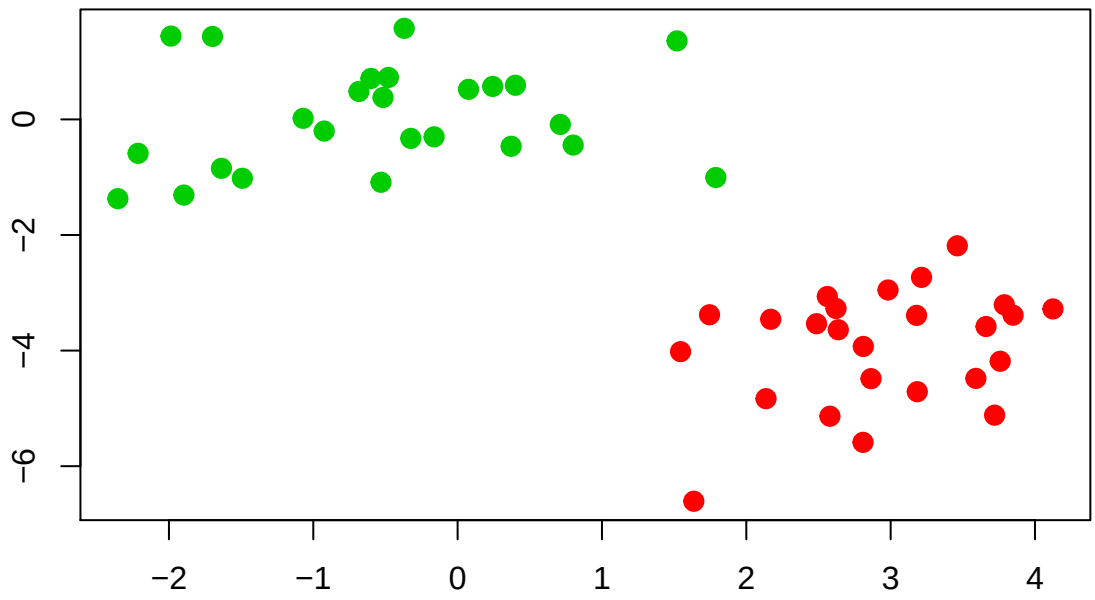
```
set.seed(2708)
x=matrix(rnorm (50*2), ncol=2)
# truly are two clusters in the data
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
km.out=kmeans(x,2) #perfectly separated the observations
km.out$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km.out$centers
```

```
##         [,1]        [,2]
## 1  2.9246016 -3.92630843
## 2 -0.5210669  0.03040455
```

```
plot(x, col=(km.out$cluster+1),
  main="K-Means Clustering Results with K=2",
  xlab="", ylab="", pch=20, cex=2)
```




```
km.out=kmeans(x,3)
km.out$cluster
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 2 2 1 1 1 1 2 1
## [36] 2 1 1 2 1 2 1 1 1 1 1 2 1 2 2
```

```
km.out$centers
```

```
##         [,1]       [,2]
## 1  0.1849663  0.2856589
## 2 -1.5801166 -0.3524770
## 3  2.9246016 -3.9263084
```

```
# variance
km.out$totss
```

```
## [1] 429.8973
```

```
km.out$withinss
```

```
## [1] 15.19079 12.89579 36.57173
```

```r
km.out$betweenss
```

```
## [1] 365.239
```

```r
sum(km.out$withinss) + km.out$betweenss
```

```
## [1] 429.8973
```

```r
km.out$betweenss/km.out$totss #how good
```

```
## [1] 0.8495959
```

```r
km.out$size #The number of points in each cluster.
```

```
## [1] 15 10 25
```

```r
km.out$iter #The number of iterations.
```

```
## [1] 2
```

```r
#Total withins ->0
#There should be no missing values
#=======================
set.seed(2708)
km.out=kmeans(x,3,nstart =20)
#The kmeans() function has an nstart option that attempts
#multiple initial configurations and reports on the best one.

km.out
```

```
## K-means clustering with 3 clusters of sizes 25, 15, 10
##
## Cluster means:
##         [,1]       [,2]
## 1  2.9246016 -3.9263084
## 2  0.1849663  0.2856589
## 3 -1.5801166 -0.3524770
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 2 2 2 2 3 2
## [36] 3 2 2 3 2 3 2 2 2 2 2 3 2 3 3
##
## Within cluster sum of squares by cluster:
## [1] 36.57173 15.19079 12.89579
##  (between_SS / total_SS =  85.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

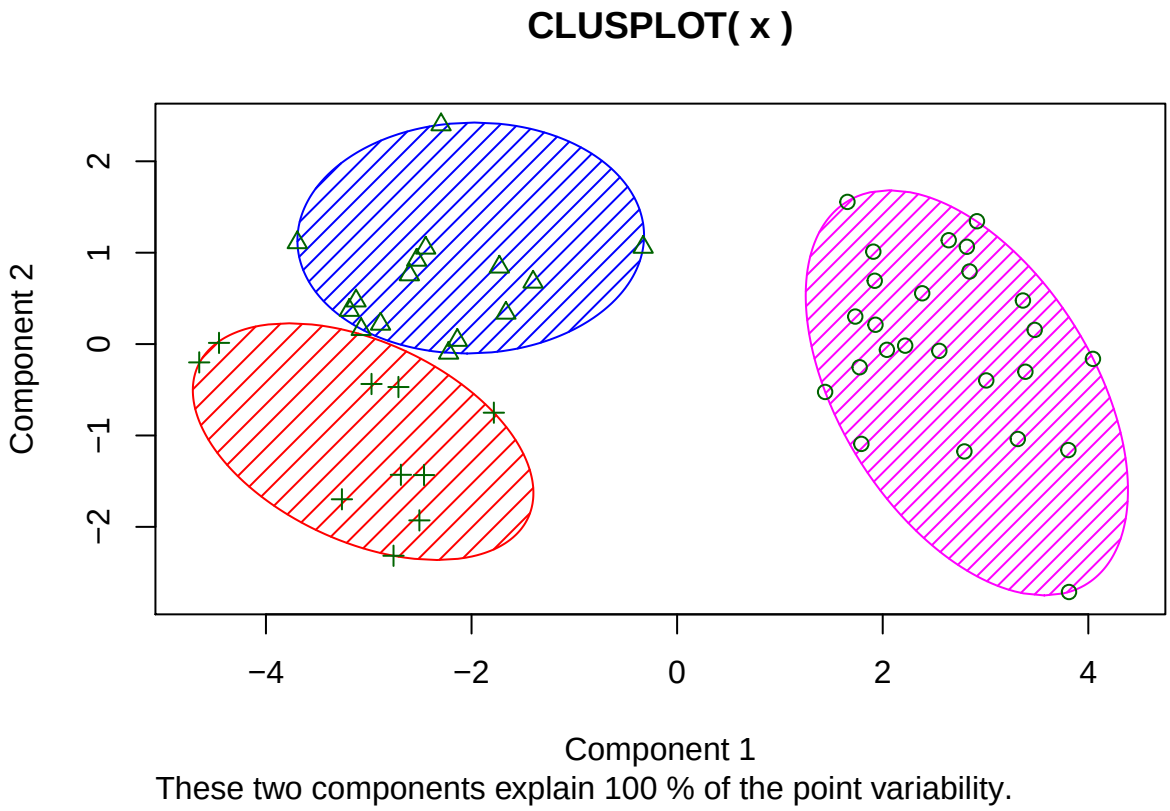
```r
set.seed(2708)
km.out=kmeans(x,3,nstart =1)
km.out$tot.withinss [1]
```

```
## [1] 68.65732
```

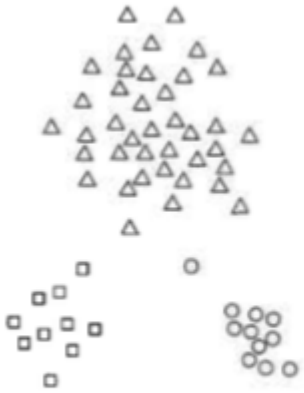
```r
km.out=kmeans(x,3,nstart =20)
km.out$tot.withinss
```

```
## [1] 64.65832
```

```r
cluster::clusplot(x, km.out$cluster, color = T, shade = T,  lines = 0)
```

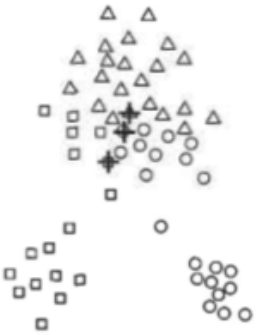


- **Choosing initial K-s**

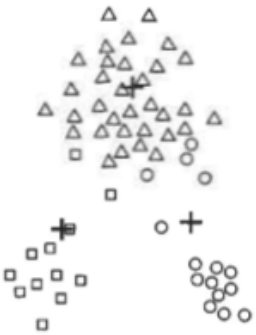


(a) Optimal clustering.

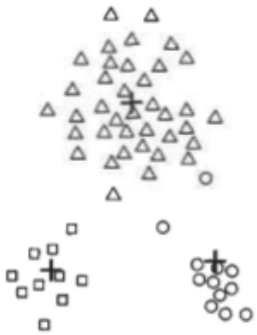
(b) Suboptimal clustering.



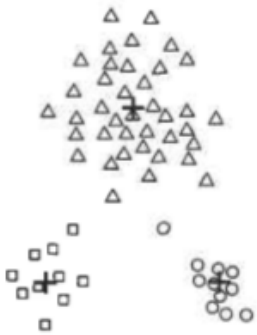
(a) Iteration 1.



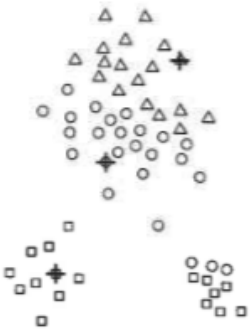
(b) Iteration 2.



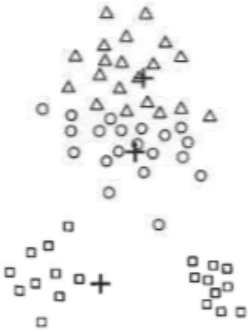
(c) Iteration 3.



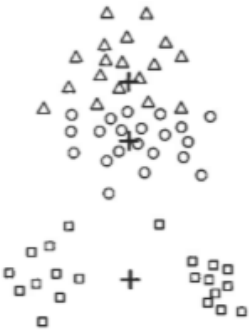
(d) Iteration 4.



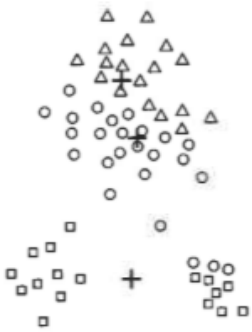
(a) Iteration 1.



(b) Iteration 2.



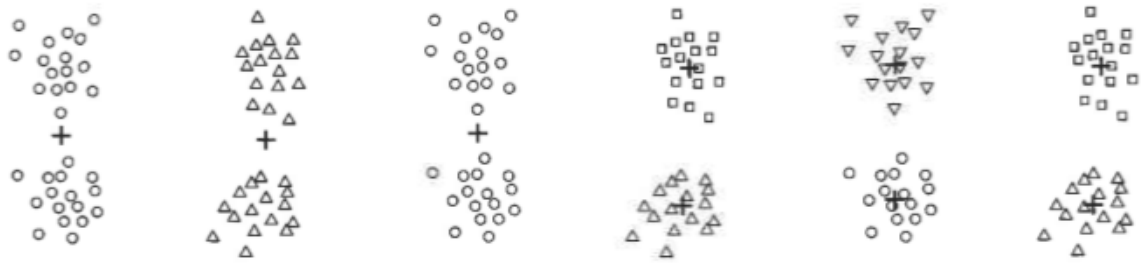
(c) Iteration 3.



(d) Iteration 4.

How to choose k

- One effective approach is to take a sample of points and cluster them using a **hierarchical clustering** technique.
- This approach often works well, but is practical only if (1) the sample is relatively small, a few hundred to a few thousand (hierarchical clustering is expensive), and (2) K is relatively small compared to the sample size.
- Select the point that is **farthest** from any of the initial centroids already selected (- outliers.)



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3.

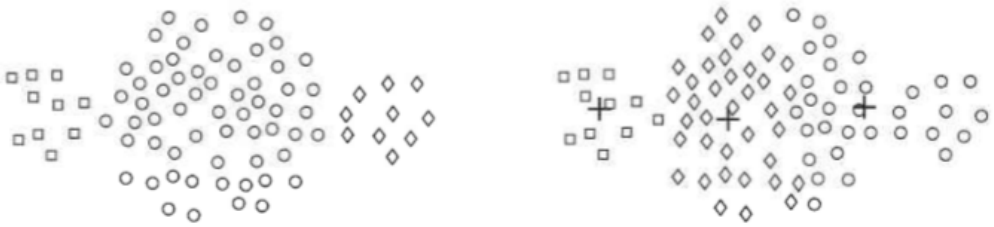
Figure 2:

Bisecting K-means

Bisecting K-means has less trouble with initialization because it performs several trial bisections and takes the one with the lowest SSE, and because there are only two centroids at each step

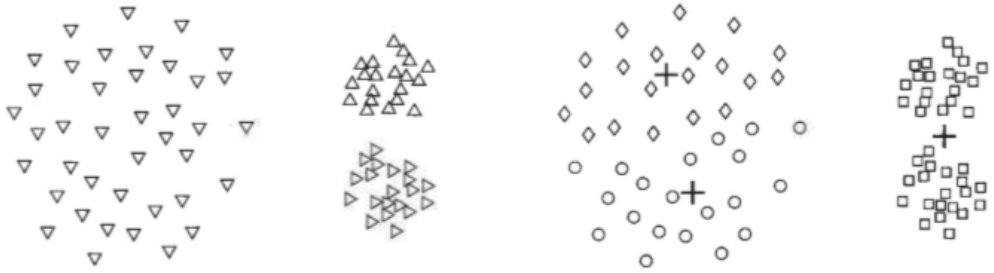
K-means and Different Types of Clusters

- **Different sizes**
- **Different densities**
- **non-spherical shapes**



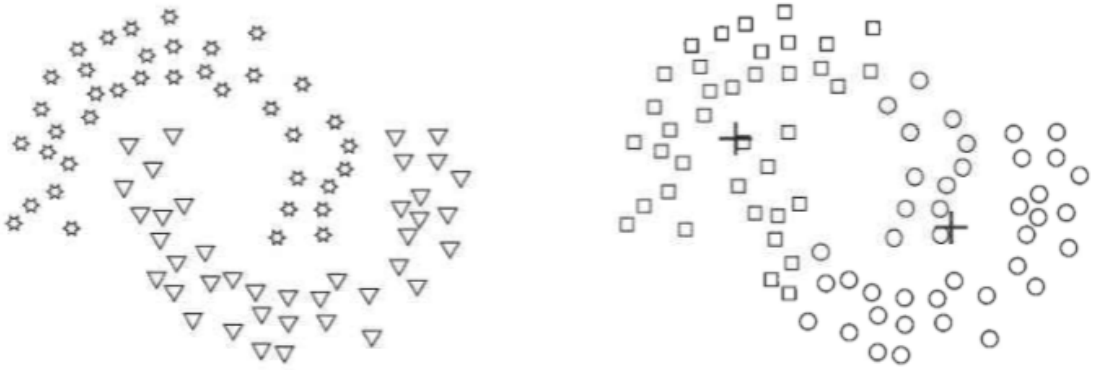
(a) Original points. (b) Three K-means clusters.

Figure 8.9. K-means with clusters of different size.



(a) Original points. (b) Three K-means clusters.

Figure 8.10. K-means with clusters of different density.



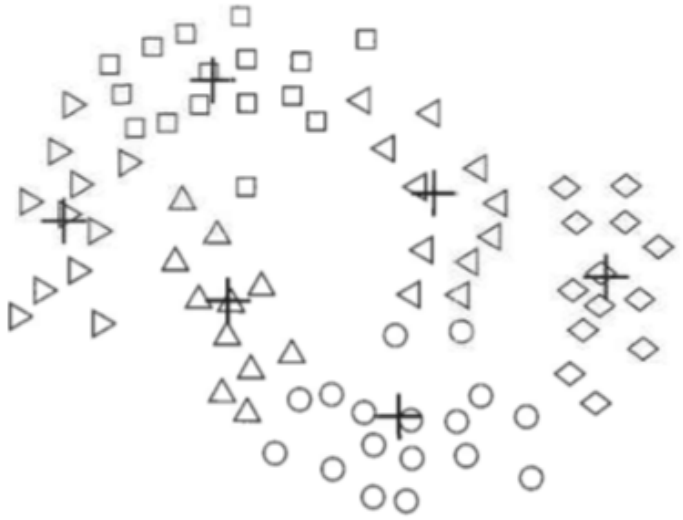
(a) Original points. (b) Two K-means clusters.

Figure 8.11. K-means with non-globular clusters.

Limitations can be overcome

(a) Unequal sizes.

(b) Unequal densities.



(c) Non-spherical shapes.

Each smaller cluster is pure in the sense that it contains only points from one of the natural clusters

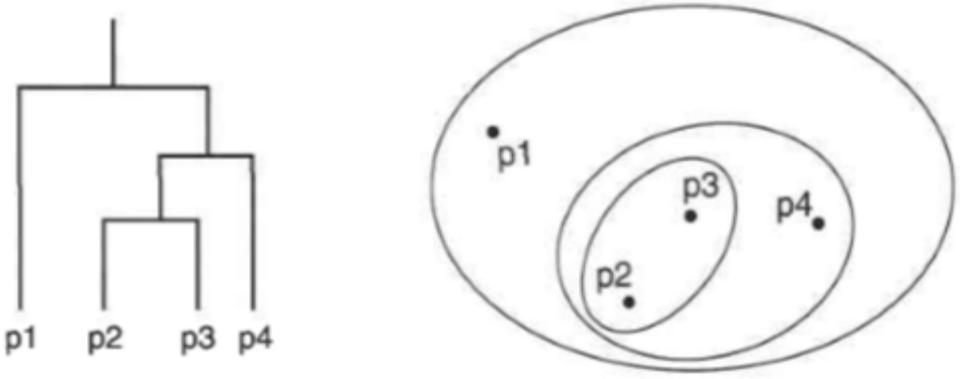
Goodness of clustering structure

- **Unsupervised**
- **Supervised**
- **Relative** (both supervised and unsupervised)
- By adding new features
- Different means
- By using different algorithms

Disadvantages

- Sensitive to outliers
- Sensitive to initial points
- Required k

One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K.



(a) Dendrogram. (b) Nested cluster diagram.

Figure 3:

Hierarchical Clustering

In K-means clustering, we seek to partition the observationsinto a pre-specified number of clusters. On the other hand, in hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram.

- One single dendrogram can be used to obtain any number of clusters

1) Agglomerative (cluster proximity, bottom-up)
2) Divisive (which cluster to split, how)

Each **leaf of the dendrogram** represents one of the n observations. However, as we move up the tree, some leaves begin to fuse into branches. observations that fuse later (near the top of the tree) can be quite different. The **height** of this fusion, as measured on the vertical axis, indicates how different the two observations are.

Problem

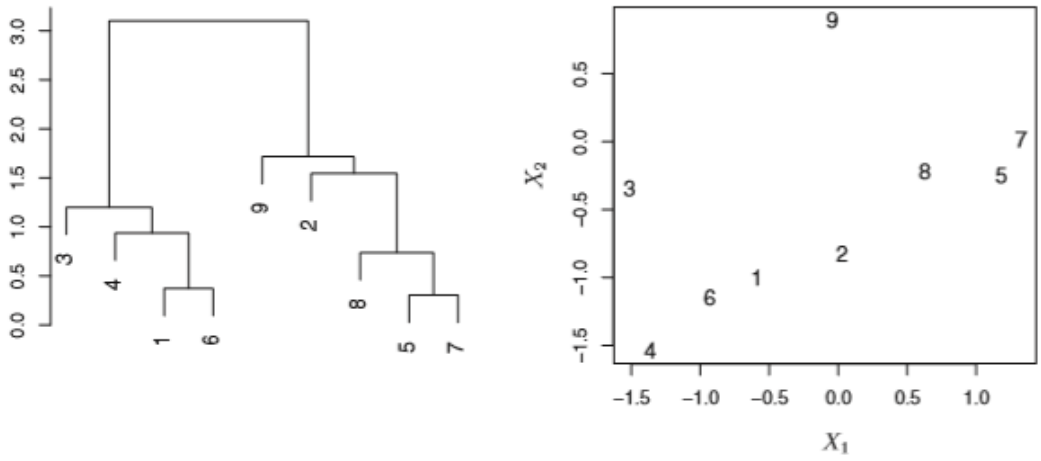


Figure 4:

Aglomerative

The algorithm proceeds *iteratively*. Starting out at the bottom of the dendrogram, each of the n observations is treated as its own cluster. Then fusing the observations. The algorithm proceeds in this fashion until all of the observations belong to one single cluster, and the dendrogram is complete.

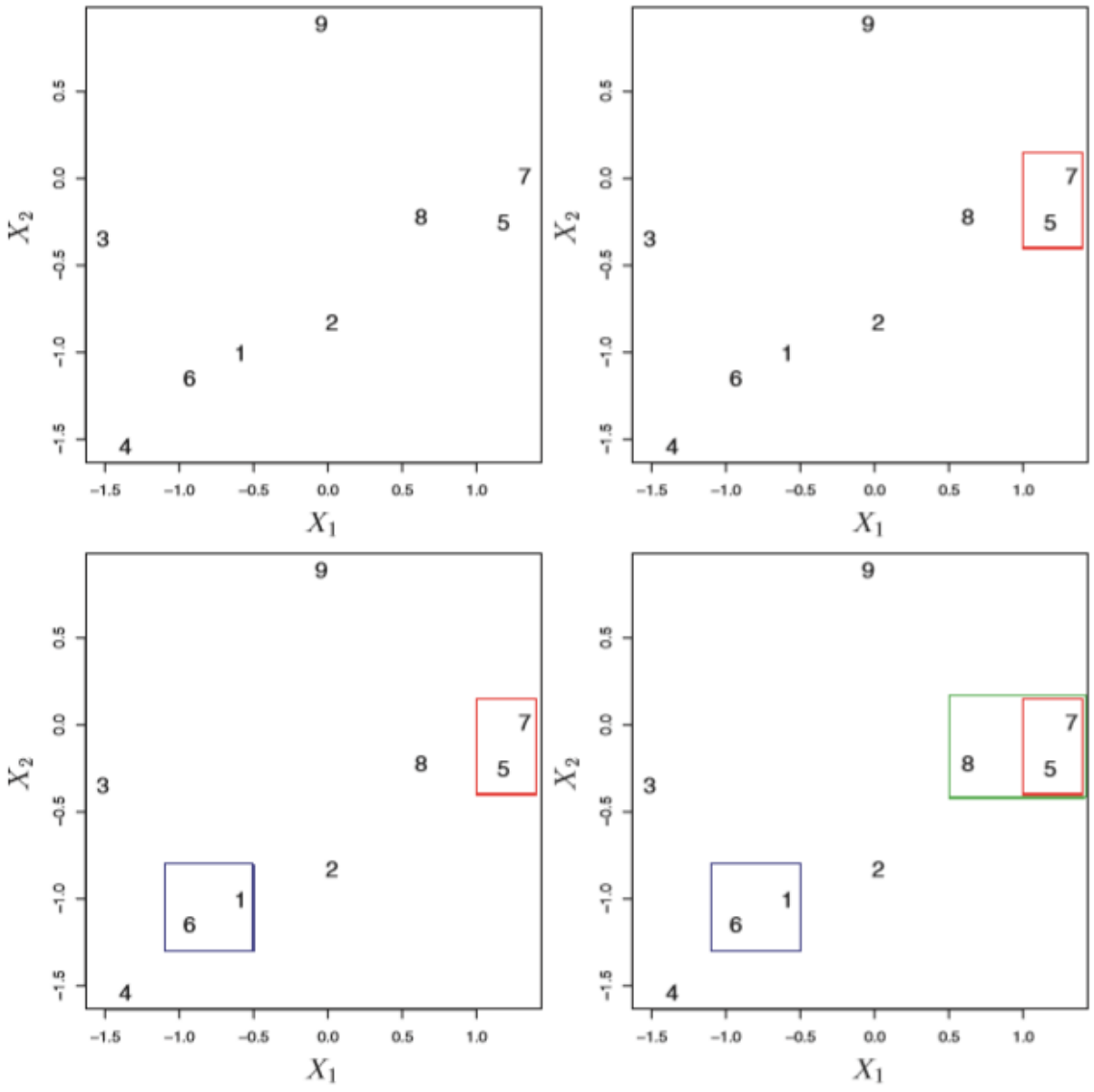
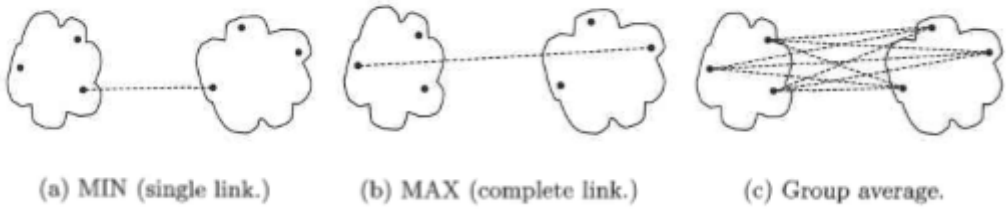


Figure 5:

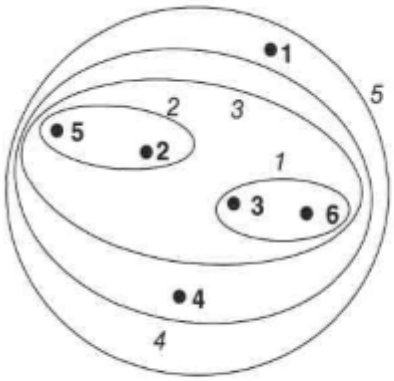


(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

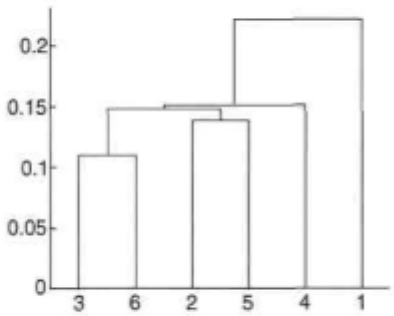
Figure 6:

Linkage

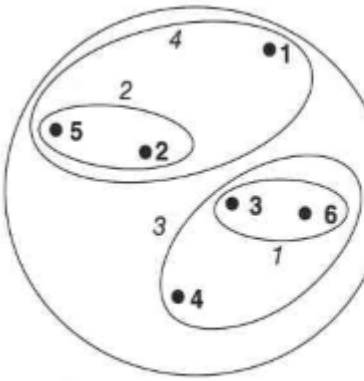
- Distance between records
- Distance between clusters (sigle link, complete link, average, centroid)



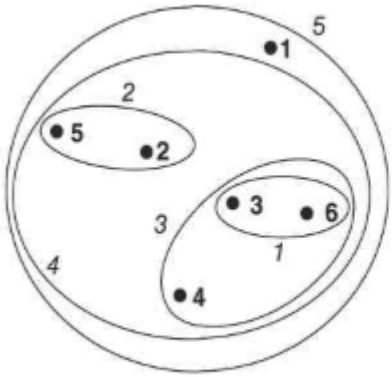
(a) Single link clustering.



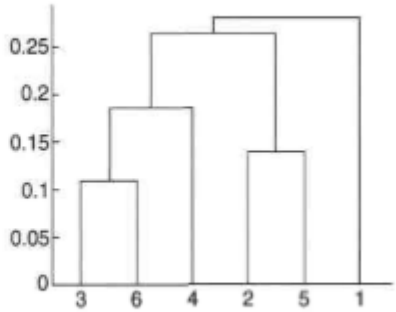
(b) Single link dendrogram.



(a) Complete link clustering.



(a) Group average clustering.



(b) Group average dendrogram.

```r
# Agglomerative method
index <- read.csv("index2017.csv")
str(index)
```

```
## 'data.frame':    186 obs. of  32 variables:
##  $ CountryID                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Country.Name                 : Factor w/ 186 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Abbr                         : Factor w/ 186 levels "AFG","AGO","ALB",..: 1 3 48 2 5 6 7 8 9 17 .
##  $ Region                       : Factor w/ 5 levels "Americas","Asia-Pacific",..: 2 3 4 5 1 3 2 3 2
##  $ World.Rank                   : int  163 65 172 165 156 33 5 30 68 90 ...
##  $ Region.Rank                  : int  40 30 14 41 26 19 4 17 15 19 ...
##  $ X2017.Score                  : num  48.9 64.4 46.5 48.5 50.4 70.3 81 72.3 63.6 61.1 ...
##  $ Property.Rights              : num  12.6 54 38.2 36.4 32.4 55.5 81.7 86 50.5 45.3 ...
##  $ Judical.Effectiveness        : num  28.4 28.5 29.6 19.8 39.6 42.5 92.9 81.8 33 48.7 ...
##  $ Government.Integrity         : num  27.5 39.7 31.7 12.8 38.2 43.4 74.8 75.2 37.6 38.2 ...
##  $ Tax.Burden                   : num  91.6 86.9 81.1 87.7 62.6 83.7 63.2 50.3 87.7 97.1 ...
##  $ Gov.t.Spending               : num  79.9 72.5 51 58.6 54.6 81.7 59 19.3 57.5 83.8 ...
##  $ Fiscal.Health                : num  97.3 51.5 19.8 70.7 56.4 82.9 84.6 79.7 97.4 42.3 ...
##  $ Business.Freedom             : num  54.2 79.3 62.1 58.5 57.3 78.5 89.3 76.9 71.5 68.5 ...
##  $ Labor.Freedom                : num  59.9 50.7 49.5 40.4 46.1 72.4 84.1 67.6 75 71.5 ...
##  $ Monetary.Freedom             : num  69.3 81.4 67 70.6 50.9 72.8 86.4 83.4 73.6 77 ...
##  $ Trade.Freedom                : num  66 87.7 63.3 56.7 66.7 80.2 86.2 87 74.4 50.6 ...
##  $ Investment.Freedom           : int  0 70 35 30 50 80 80 90 55 50 ...
##  $ Financial.Freedom            : int  0 70 30 40 50 70 90 70 50 60 ...
##  $ Tariff.Rate                  : num  7 1.1 8.4 11.7 6.6 2.4 1.9 1.5 5.3 19.7 ...
##  $ Income.Tax.Rate              : num  20 23 35 17 35 26 45 50 25 0 ...
##  $ Corporate.Tax.Rate           : num  20 15 23 30 35 20 30 25 20 0 ...
##  $ Tax.Burden.perc.of.GDP       : num  6.5 23.6 11.7 6.5 35.9 23.5 27.5 43 14.2 16.9 ...
##  $ Gov.t.Expenditure.perc.of.GDP: num  27.1 30 44.4 28.9 43.9 26.4 37.2 51.9 38.5 24.2 ...
##  $ Population_Millions          : num  32 2.8 39.5 25.1 42.4 3.3 23.9 8.6 9.5 0.4 ...
##  $ GDP.Billions.PPP             : num  62.3 32.7 578.7 184.4 972 ...
##  $ GDP.Growth.Rate              : num  1.5 2.6 3.7 3 1.2 3 2.5 0.9 1.1 0.5 ...
##  $ GDP.per.Capita.PPP           : int  1947 11301 14504 7344 22554 8468 47389 47250 17993 25167 ...
##  $ Unemployment                 : num  9.6 17.3 10.5 7.6 6.7 16.3 6.3 5.7 4.7 14.4 ...
##  $ Inflation.Perc               : num  -1.5 1.9 4.8 10.3 26.5 3.7 1.5 0.8 4 1.9 ...
##  $ FDI.Inflow.Millions          : num  58 1003 -587 8681 11655 ...
##  $ Public.Debt.Perc.of.GDP      : num  6.8 71.9 8.7 62.3 56.5 46.6 36.8 86.2 36.1 65.7 ...
```

```r
rownames(index) <- index$Abbr
#chosing the subset of observations and features to show the dendogram
index1 <- index[1:7, c("Unemployment", "GDP.per.Capita.PPP")]
# Preparing the data
index1 <- na.omit(index1)
d <- dist(index1, method = "euclidian")
d
```

```
##           AFG       ALB       DZA       AGO       ARG       ARM
## ALB  9354.003
## DZA 12557.000  3203.007
## AGO  5397.000  3957.012  7160.001
## ARG 20607.000 11253.005  8050.001 15210.000
## ARM  6521.003  2833.000  6036.003  1124.034 14086.003
## AUS 45442.000 36088.002 32885.000 40045.000 24835.000 38921.001
```

```r
cl <- hclust(d, method = "complete")
cl
```

```
##
## Call:
## hclust(d = d, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 7
```

```r
cl$merge # - observation, + group
```

```
##      [,1] [,2]
## [1,]   -4   -6
## [2,]   -2   -3
## [3,]   -1    1
## [4,]   -5    2
## [5,]    3    4
## [6,]   -7    5
```

```r
index1
```

```
##     Unemployment GDP.per.Capita.PPP
## AFG          9.6               1947
## ALB         17.3              11301
## DZA         10.5              14504
## AGO          7.6               7344
## ARG          6.7              22554
## ARM         16.3               8468
## AUS          6.3              47389
```

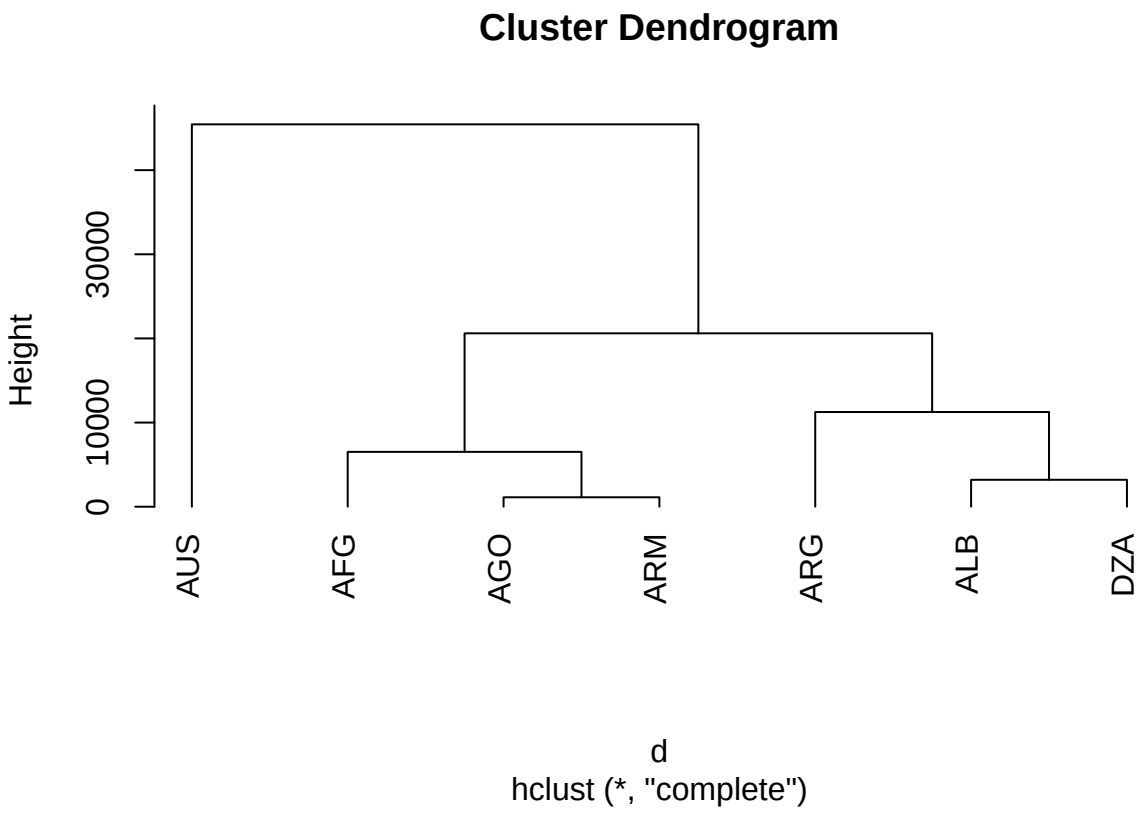
```r
d #1124
```

```
##           AFG       ALB       DZA       AGO       ARG       ARM
## ALB  9354.003
## DZA 12557.000  3203.007
## AGO  5397.000  3957.012  7160.001
## ARG 20607.000 11253.005  8050.001 15210.000
## ARM  6521.003  2833.000  6036.003  1124.034 14086.003
## AUS 45442.000 36088.002 32885.000 40045.000 24835.000 38921.001
```

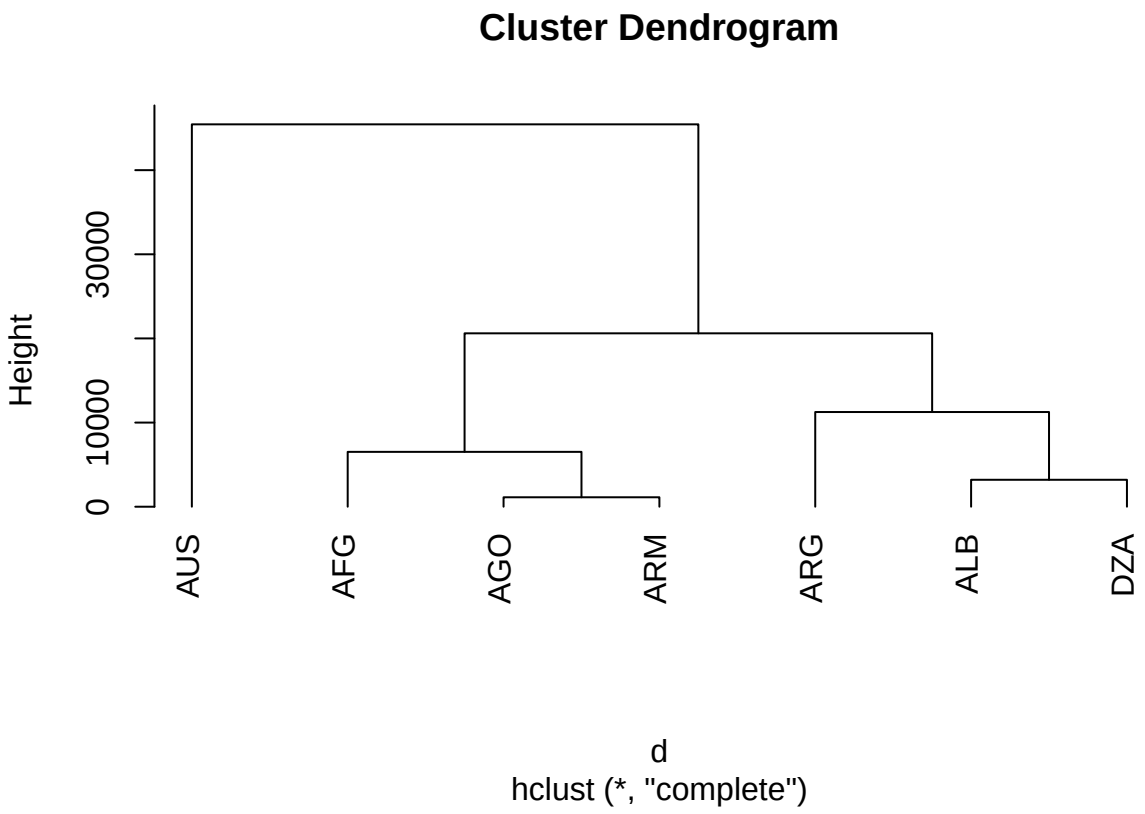
```r
cl$height
```

```
## [1]  1124.034  3203.007  6521.003 11253.005 20607.000 45442.000
```

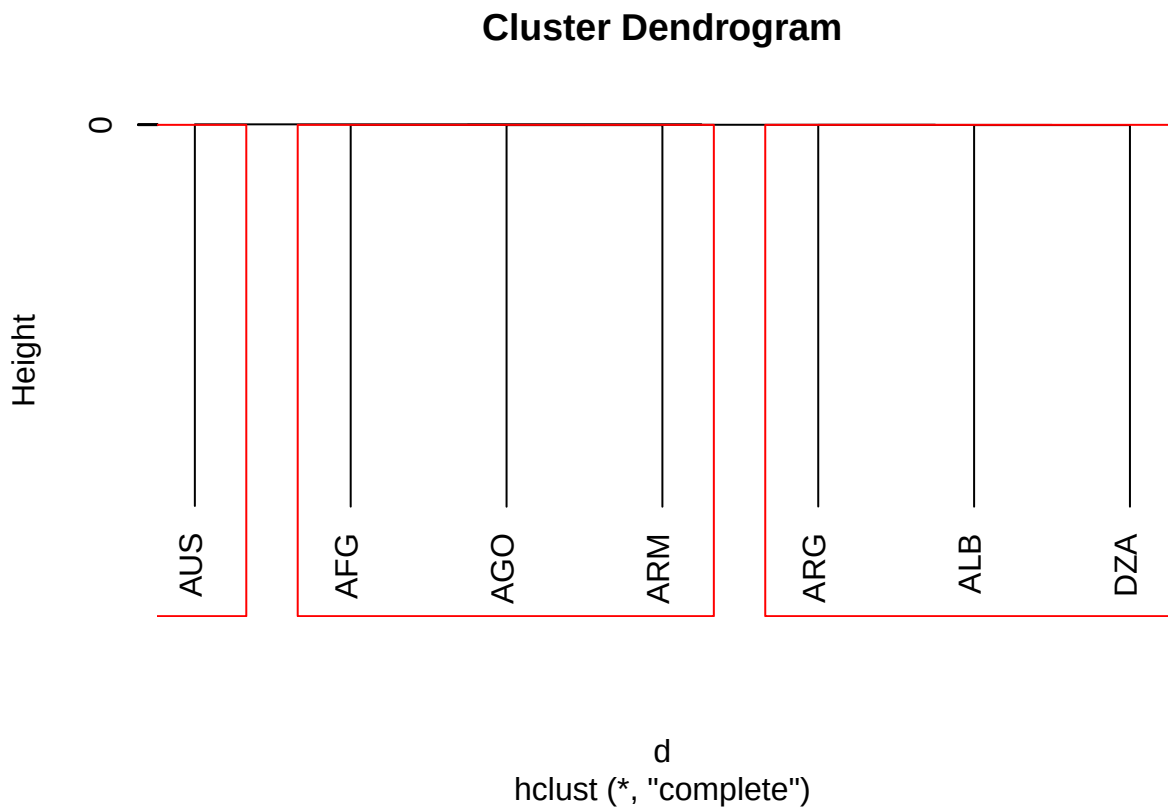
```r
plot(cl, hang = -1)
```



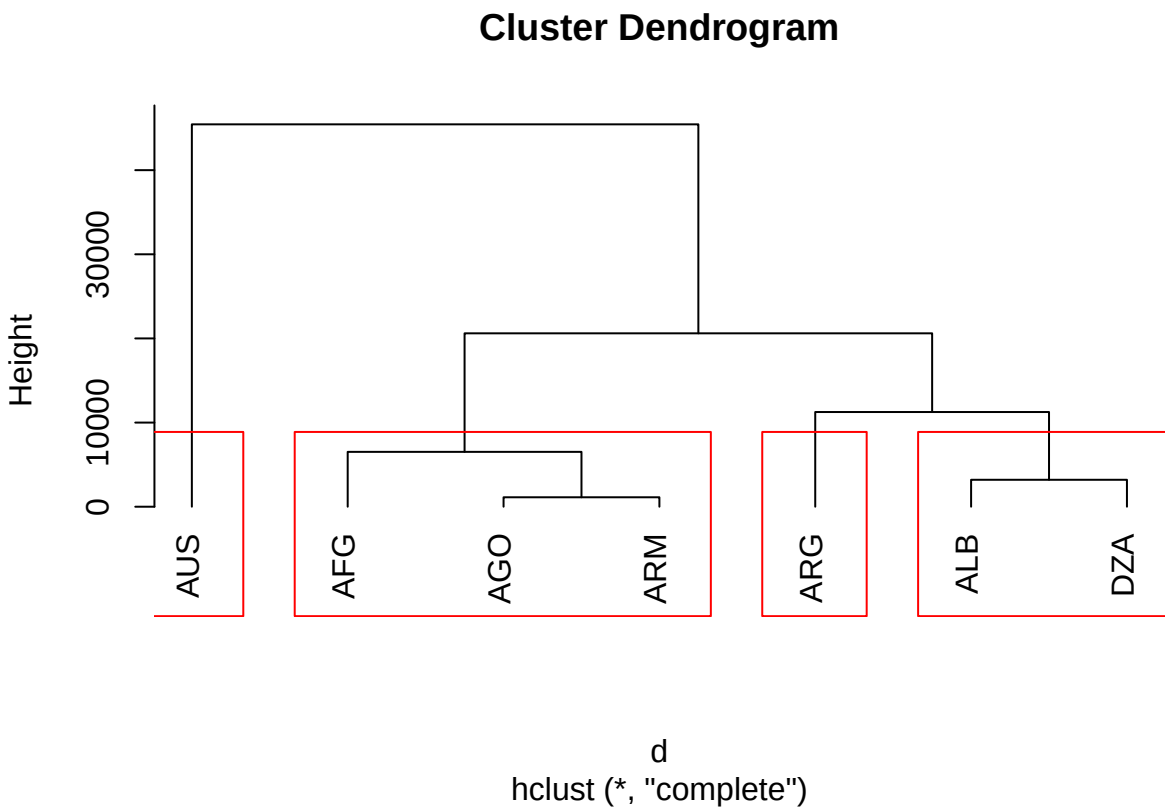

```
plot(cl, hang = -500)
```



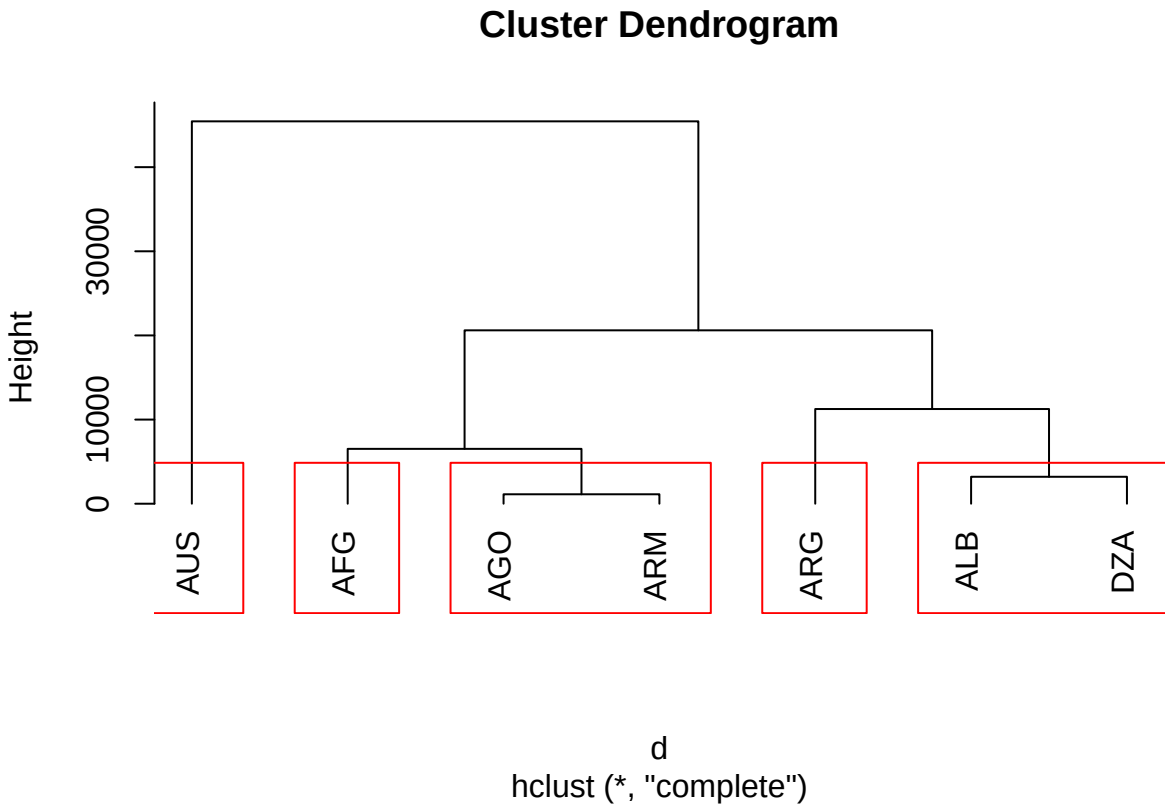

```
plot(cl, hang = 500)
rect.hclust(cl,3)
```




```
plot(cl, hang = -1)
rect.hclust(cl,4)
```



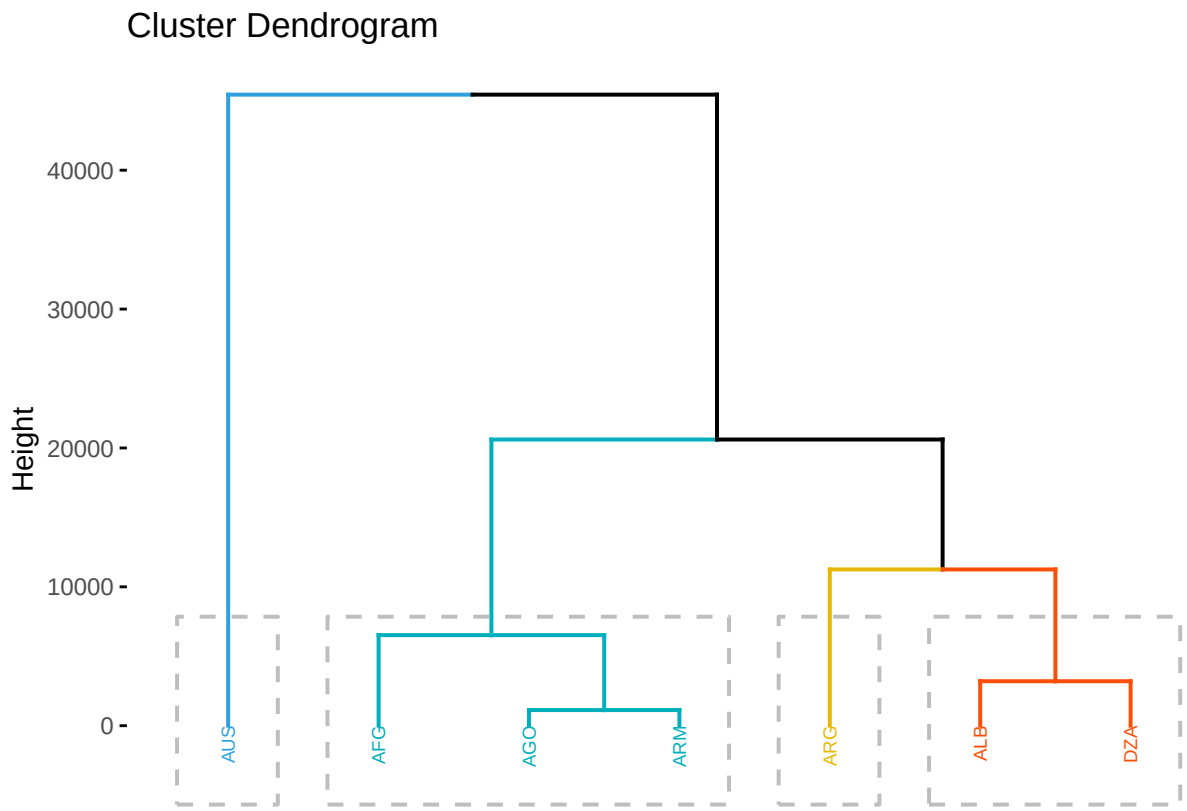

```
plot(cl, hang = -1)
rect.hclust(cl,5)
```




```
index1$cl.memb <- cutree(cl, k=3)
index1
```

```
##     Unemployment GDP.per.Capita.PPP cl.memb
## AFG          9.6               1947       1
## ALB         17.3              11301       2
## DZA         10.5              14504       2
## AGO          7.6               7344       1
## ARG          6.7              22554       2
## ARM         16.3               8468       1
## AUS          6.3              47389       3
```

```
factoextra::fviz_dend(cl, k = 4, # Cut in four groups
          cex = 0.5, # label size
          k_colors = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
          color_labels_by_k = TRUE, # color labels by groups
          rect = TRUE # Add rectangle around groups
          )
```

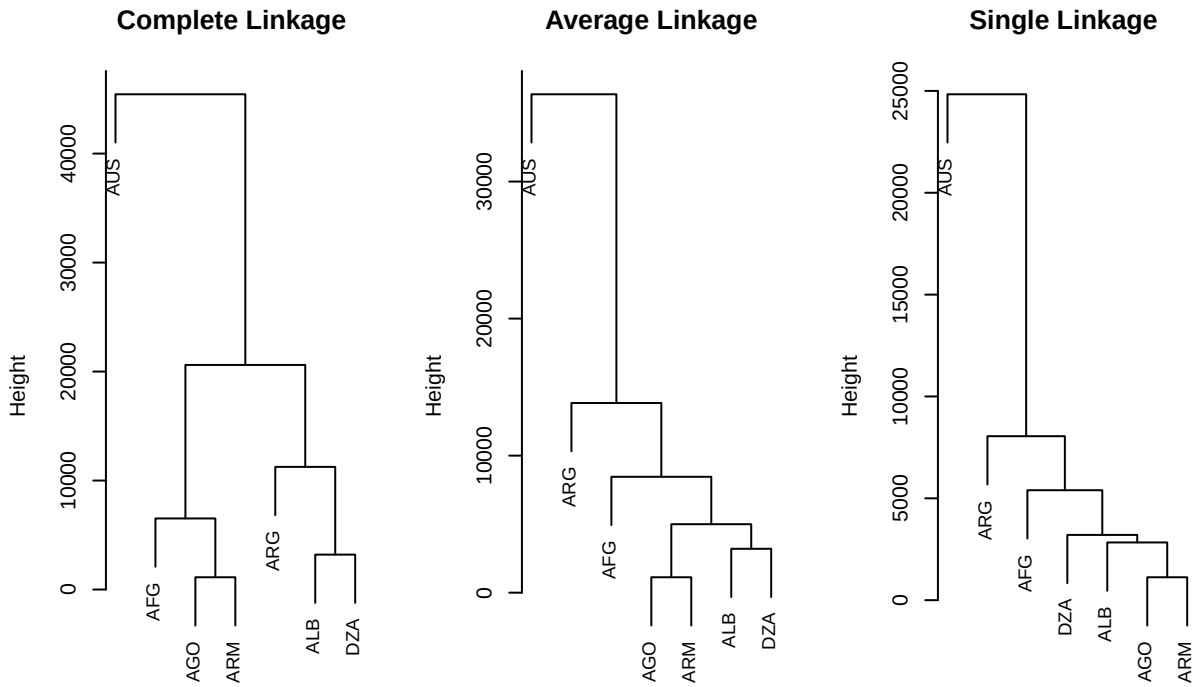



```
hc.complete =hclust(dist(index1), method="complete")
hc.average =hclust(dist(index1), method ="average")
hc.single=hclust(dist(index1), method ="single")

par(mfrow=c(1,3))
plot(hc.complete ,main="Complete Linkage ", xlab="", sub="", cex=.9)

plot(hc.average , main="Average Linkage", xlab="", sub="", cex=.9)

plot(hc.single, main="Single Linkage ", xlab="", sub="", cex=.9)
```




```
hc.single$merge
```

```
##      [,1] [,2]
## [1,]   -4   -6
## [2,]   -2    1
## [3,]   -3    2
## [4,]   -1    3
## [5,]   -5    4
## [6,]   -7    5
```

```
#To determine the cluster labels
cutree(hc.complete , 2)
```

```
## AFG ALB DZA AGO ARG ARM AUS
##   1   1   1   1   1   1   2
```

```
cutree(hc.average , 2)
```

```
## AFG ALB DZA AGO ARG ARM AUS
##   1   1   1   1   1   1   2
```

```
cutree(hc.single , 2)
```

```
## AFG ALB DZA AGO ARG ARM AUS
##   1   1   1   1   1   1   2
```

```
cutree(hc.single , 4)
```

```
## AFG ALB DZA AGO ARG ARM AUS
##   1   2   2   2   3   2   4
```

```
# The problem is scaling
par(mfrow=c(1,1))
par(mfrow=c(1,2))
index2 <- scale(index1)
plot(hclust(dist(index2), method ="complete"), main="Hierarchical Clustering with Scaled Features")
plot(hclust(dist(index1), method ="complete"), main="Hierarchical Clustering without Scaled Features")
```