Unsupervised Learning - Clustering

Mark Asch - IMU/VLP/CSU

2023

Program

1. Data Analysis
   (a) Introduction: the 4 identifiers of "big data" and "data science"
   (b) Supervised learning methods: regression—advanced, k-NN, linear classification methods, SVM, NN, decision trees.
   (c) Unsupervised learning methods: principal component analysis, k-means, clustering.

Recall: Introduction

- Supervised learning:
  - ⇒ we have p characteristics $X_1, X_2, \ldots, X_p$ measured on n observations
  - ⇒ one response Y measured on the same observations
  - ⇒ objective: predict Y using $X_1, X_2, \ldots, X_p$
  - ⇒ methods: regression and classification

- Unsupervised learning:
  - ⇒ we only have p characteristics $X_1, X_2, \ldots, X_p$ measured on n observations
  - ⇒ we want to make predictions, but we do not have an associated response variable...
  - ⇒ objective: discover interesting effects with respect to the observations of $X_1, X_2, \ldots, X_p$
    - → Can we visualize or represent the data in a more informative way?
    - → Can we discover sub-groups or clusters among the variables or the observations?

Recall: Warnings!

- Unsupervised learning is more difficult, since it is subjective.

✘ Unsupervised learning is often part of an initial phase of exploratory data analysis (EDA), where we compute elementary statistics and plot basic histograms and boxplots---see Introductory lectures.

✘ There are no universal cross-validation methods for unsupervised learning.

✘ There is no response variable that can be used to test our models.

✔ However, there are a large number of application domains where unsupervised learning is widely used.

Introduction to Clustering

- Clustering is an ensemble of methods for finding subgroups, or clusters, in a database.

- The subgroups are defined according to two principles:

  $\Rightarrow$ The observations within each group are similar.
  $\Rightarrow$ The observations in different groups are different.

- For example, suppose we have n observations, each with p properties.

  $\Rightarrow$ The properties correspond to measurements made for each sample among the n.
  $\Rightarrow$ The clustering will enable us to distinguish between subtypes within the measurements
  $\Rightarrow$ The problem is unsupervised because we are attempting to discover the unknown structure, based exclusively on the measured data.

Clustering Approaches

There are a large number of clustering methods. The two best known are:

1. k-means clustering.

2. Hierarchical, or tree clustering.

The major difference is that:

- in k-means we try to divide the observations into a specified number of clusters,

- whereas in the hierarchical approach we do not know in advance the number of clusters, and we end up with a tree-like structure known as a *dendrogram*. This structure enables us to visualize all the possible clusterings, from 1 to n.

- **Note**:

- $\Rightarrow$ both PCA and clustering seek to simplify the data by means of a small number of parameters, but their mechanisms are different:
- $\Rightarrow$ PCA tries to find a low dimensional representation of the observations that explains a large proportion of the variance.
- $\Rightarrow$ Clustering tries to find homogeneous subgroups among the observations.

K-MEANS CLUSTERING

k-means Clustering

- This is a simple and elegant method for dividing a dataset into k distinct clusters with no overlaps.
- To do the clustering, one must first specify the number of desired clusters, K.
- The algorithm will then assign each observation to at most one of the K clusters.

$\Rightarrow$ Since we do not know the value of k, the algorithm will be based on an optimization problem.

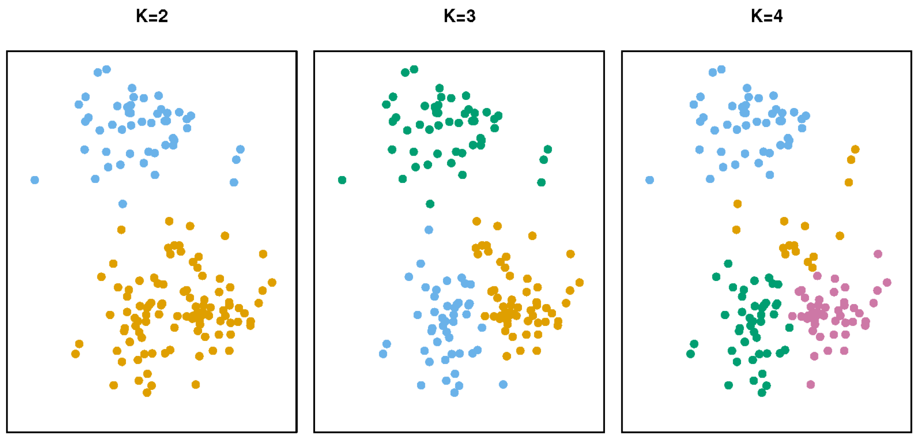


Figure 1: Application of k-means to simulated 2-D data.

Theory of k-means

- Let $C_1, \ldots, C_K$ be the sets containing the indices of the observations in each of the K clusters.

- These sets satisfy two properties:

  1. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \ldots, n\}$, so each observation belongs to at least one of the K clusters.
  2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$, meaning that the clusters do not overlap and no observation can belong to more than one cluster.

- Then a "good" clustering is defined as one for which the intra-cluster variation, $W(C_k)$, is the smallest possible. That is, we want to solve the optimization problem

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\},$$

with a suitable definition of the variation $W(C_k)$.

- There are several possibilities for this, but we usually just use the squared Euclidean distance,

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2,$$

  where $|C_k|$ is the number of observations in the k-th cluster.

- Substituting this expression, we obtain the k-means optimization problem,

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \right\}. \quad (1)$$

Algorithm for k-means

- Solving this optimization problem is not trivial.

  ⇒ How do we find a division of the observations into K clusters that guarantees the minimization of the above cost function?

- It can be proved that a very simple algorithm can find an acceptable solution.

  ⇒ (1) Assign a number between 1 and K at random to each observation--this produces the initial clustering.

  ⇒ (2) Iterate the following until the cluster assignments do not change anymore:

  → (2a) For each cluster, compute the centroid equal to the vector of the p characteristic means for all observations in cluster p.

  → (2b) Assign each observation to the cluster whose centroid is closest as defined by the chosen distance function.

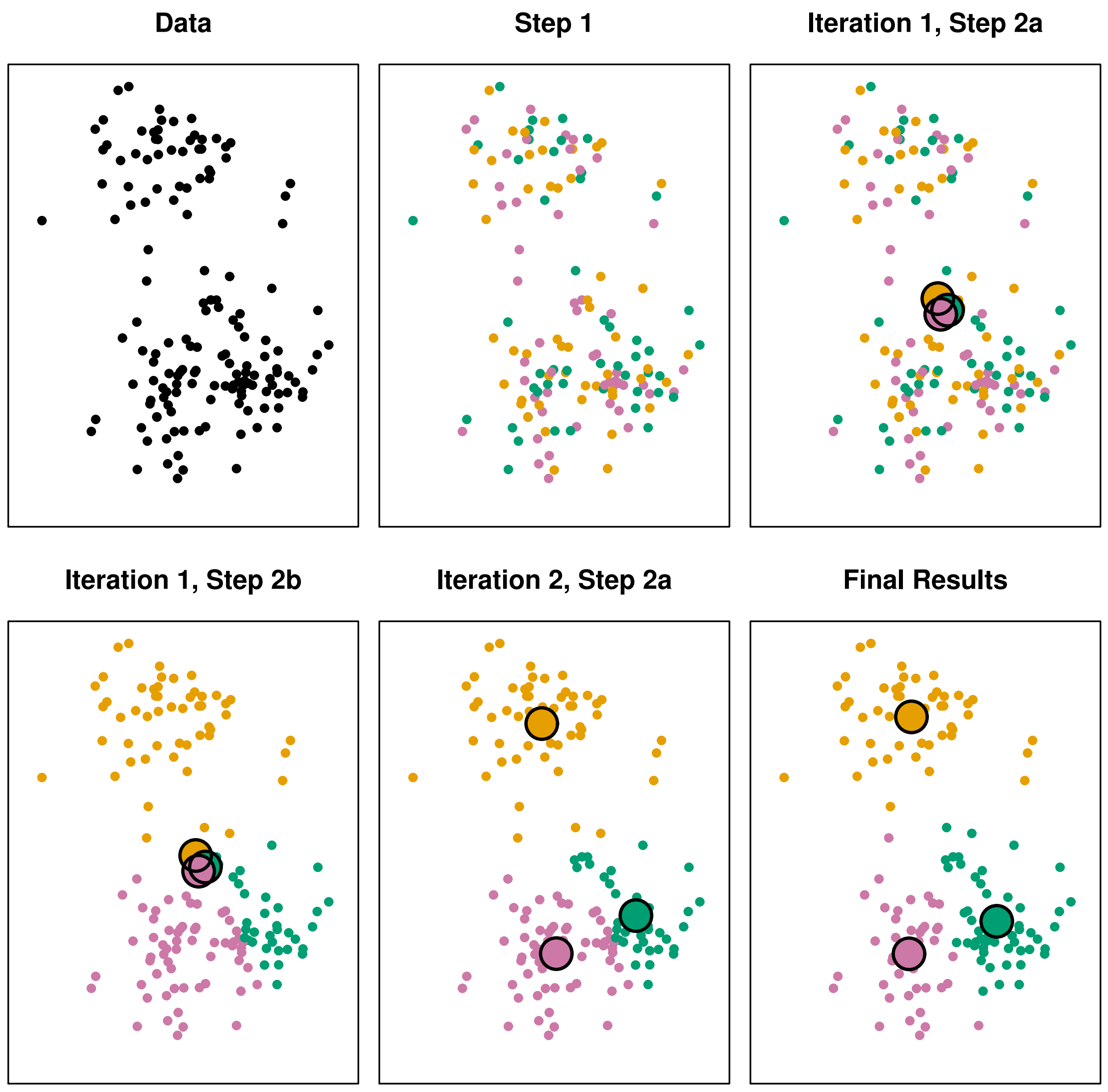


Figure 2: Progress of the k-means algorithm

✔ Since this algorithm finds a local optimum, the results will depend on the initial, random assignment of labels.

- ✔ It is thus vital to relaunch the algorithm a large number of times, starting from different random initial configurations.
- ✔ Finally, we choose the best solution that produces the smallest value of the cost function.

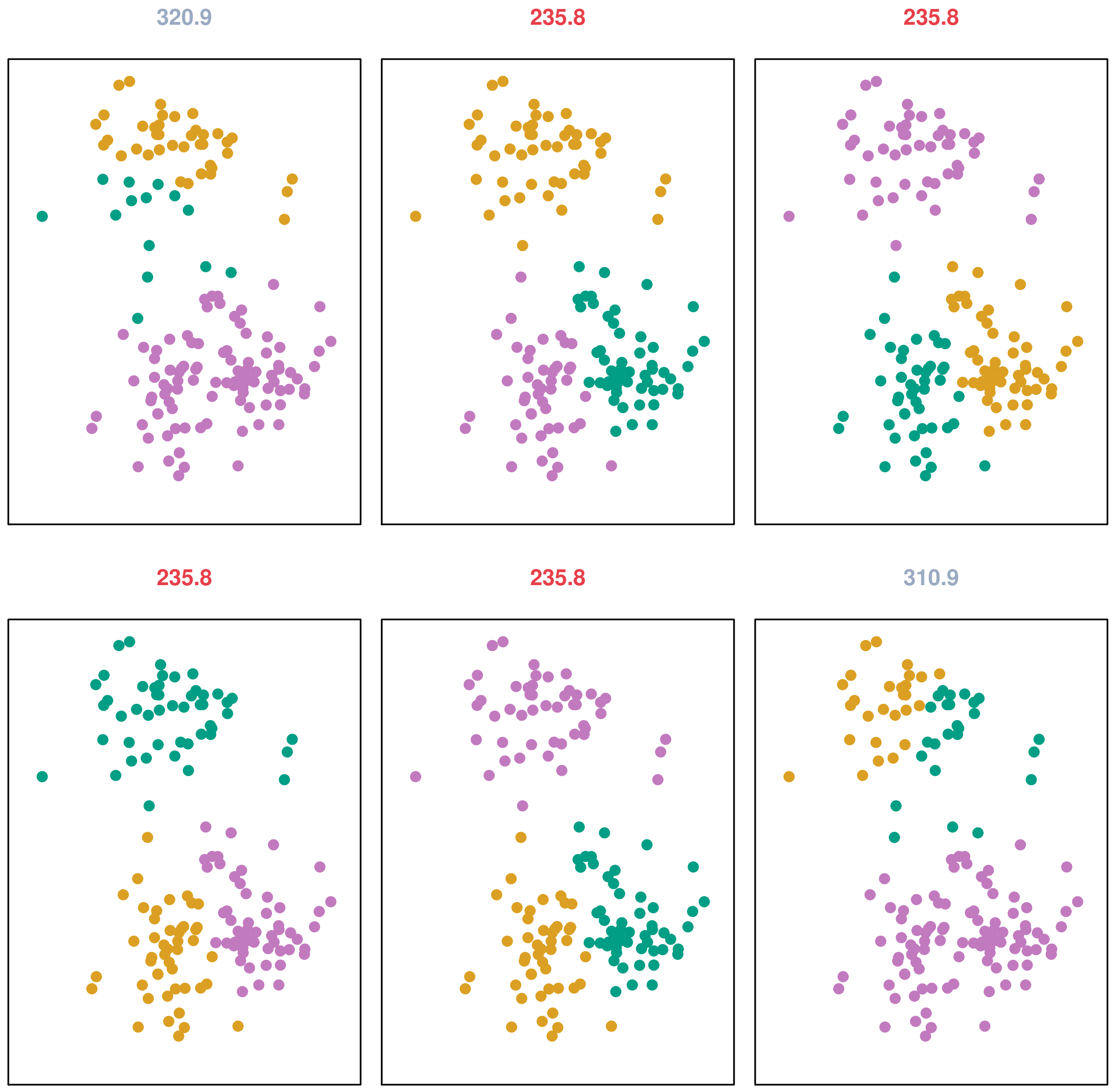


Figure 3: Six runs with $K = 3$

Practical Considerations for Clustering

- Clustering is a very useful tool for data analysis.
- Before starting, we must make two decisions whose potential impacts are fundamental:
  1. Should we normalize (center and reduce) the attributes?
  2. How many clusters should we seek?
- In practice we will have to try several options, since there is no single answer to these two questions.
- The problem of validating the computed clusters is a subject of ongoing research.
- Finally, to interpret the results, one can do the following:

$\Rightarrow$ Tune the parameters—see Examples below.
$\Rightarrow$ Check the robustness by recalculating over subsets of the data—a form of cross-validation.
$\Rightarrow$ Be very cautious when reporting the results—they do not represent the absolute truth!

Example

- The function `kmeans()` performs k-means clustering in `R`
- Consider a simple, simulated example: there are 2 clusters in the data, with a shift in their mean values between the first 25 and the last 25 observations

```
set.seed(2)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
head(x)
##          [,1]      [,2]
## [1,] 2.103085 -4.838287
## [2,] 3.184849 -1.933699
## [3,] 4.587845 -4.562247
## [4,] 1.869624 -2.724284
## [5,] 2.919748 -5.047573
## [6,] 3.132420 -5.965878
```

- First, perform k-means with $K = 2$:

```
km.out=kmeans(x,2,nstart=20)
```

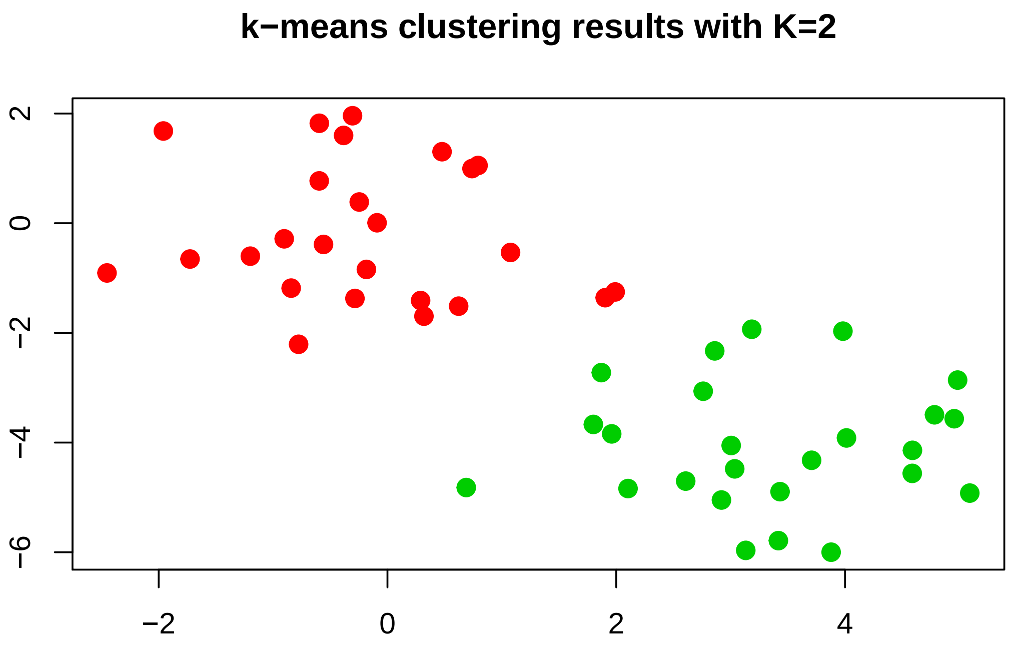
- The affectations of the observations are in the variable `cluster`:

```
km.out$cluster
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
         2 2 2 2 2 2 2 2 2 2 1 1 1 1 1
         1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

- We observe that k-means has perfectly separated the observations into two clusters, without us having supplied any group information to `kmeans()`...

- Plot the observations, colored by their cluster affectation :

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

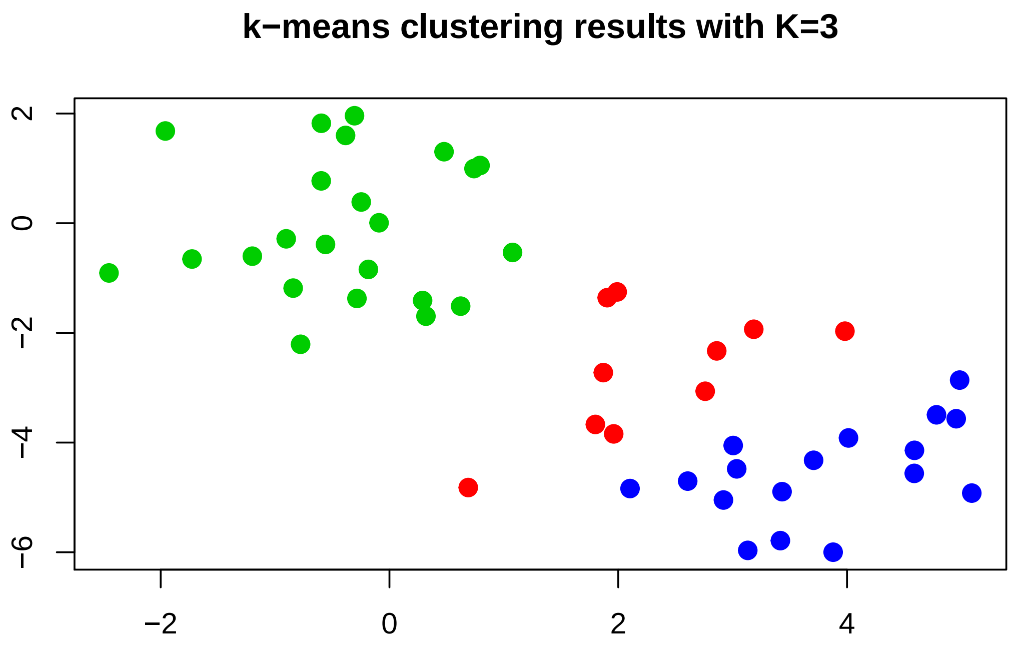


- Here, the observations are easy to plot, since we are in 2 dimensions only... In the case of more than 2 variables, we could perform a PCA and then plot the first 2 principal components.

- In this simulated example we knew the number of clusters. But in general this is definitely not the case. So we could have started by trying $K = 3$:

```
set.seed(4)
km.out=kmeans(x,3,nstart=20)
km.out
```

```
plot(x, col=(km.out$cluster+1), main="K-Means Clustering
Results with K=3", xlab="", ylab="", pch=20, cex=2)
## K-means clustering with 3 clusters of sizes 10, 23, 17
##
## Cluster means:
##         [,1]        [,2]
## 1  2.3001545 -2.69622023
## 2 -0.3820397 -0.08740753
## 3  3.7789567 -4.56200798
##
## Clustering vector:
##  [1] 3 1 3 1 3 3 3 1 3 1 3 1 3 1 3 1 3 3 3 3 3 1 3 3 3
        2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 19.56137 52.67700 25.74089
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```



- Here, the algorithm has divided one of the original 2 clusters...

- Note that to execute `kmeans()` with different initial affectations, we use the argument `nstart`. here we compare `nstart=1` with `nstart=20` by extracting the intracluster sum of squares score.

```
set.seed(3)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
## [1] 104.3319
```

```
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss
## [1] 97.97927
```

- **Conclusions**:
  - ⇒ Restarting has effectively reduced the value of the intracluster sum of squares.
  - ⇒ We have attained a better optimum (minimum).
- It is strongly recommended to execute many times, using a high value of `nstart`, such as 20 or 50, otherwise undesirable local minima will be found...
- We also use a random seed initialization, by `set.seed()`, to ensure reproducibility of the results.

HIERARCHICAL CLUSTERING

Hierarchical Clustering

- No need to choose the number of clusters beforehand.
- Everything is done " bottom-up," starting from the leaves and moving up to the trunk, or main branch
- This generates a tree-like diagram, known as a dendrogram.
- It is a deterministic method, but can provide clues for further inference in statistical models.

Interpretation

- Suppose we have synthetic data, in 2D, having 3 distinct classes, but that are observed WITHOUT any knowledge of their classes.

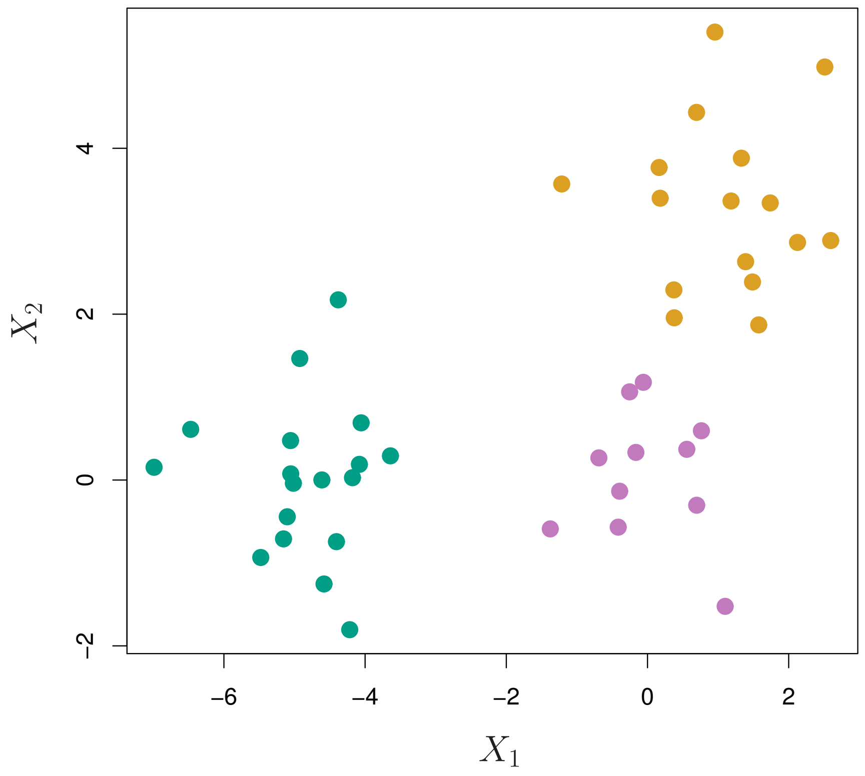


Figure 4: 45 observations in 3 classes

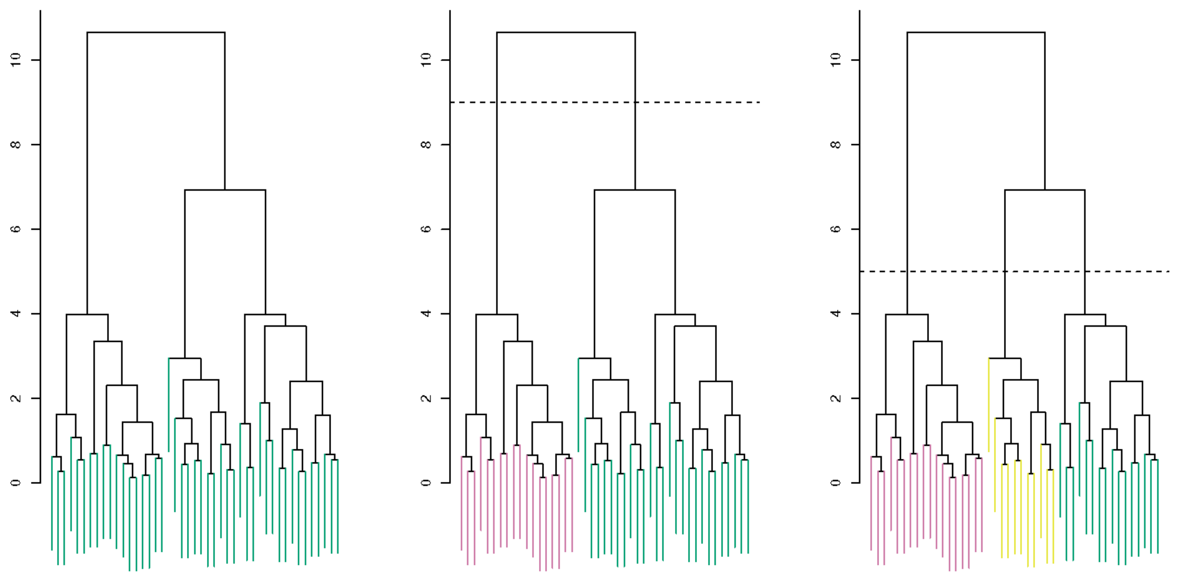


Figure 5: Dendrogram obtained by hierarchical clustering with full linkage.

- Each leaf represents one of the 45 observations (figure on left).

- Moving up, the leaves merge into branches, corresponding to observations that have a similarity---measured in a given metric---between them.

  $\Rightarrow$ The earlier the merging takes place, the more similar are the observations.

⇒ Thus, observations that merge later---towards the top---are more and more different.

- Conclusion: the merging height, on the vertical axis, indicates the difference between two observations and it is the height that will determine the number of different clusters.

- Identification of clusters:

⇒ make a horizontal cut across the dendrogram
⇒ the distinct sets below can be interpreted as distinct clusters
⇒ further cuts can be made, going downwards, until each observation is in its own cluster... (this is not very useful!)

- A single dendrogram can provide any number of clusters, from 1 to N, the number of observations.

⇒ Choosing the "good" number of clusters, or cut height, is not an obvious task, and has to be performed visually, based on experience, or familiarity with the data

⇒ Hopefully, however, there is a clear gap in the link lengths of the dendrogram that will separate the inherent clusters from the unnatural ones. This will depend on the context and the data.

- The dendrogram does not work better than K-means, unless the data are truly hierarchical... (without other divisions, eg. male-female)

Algorithm for hierarchical clustering

1. For all n observations, calculate the $n(n-1)/2$ dissimilarities pairwise. Each observation is a cluster.

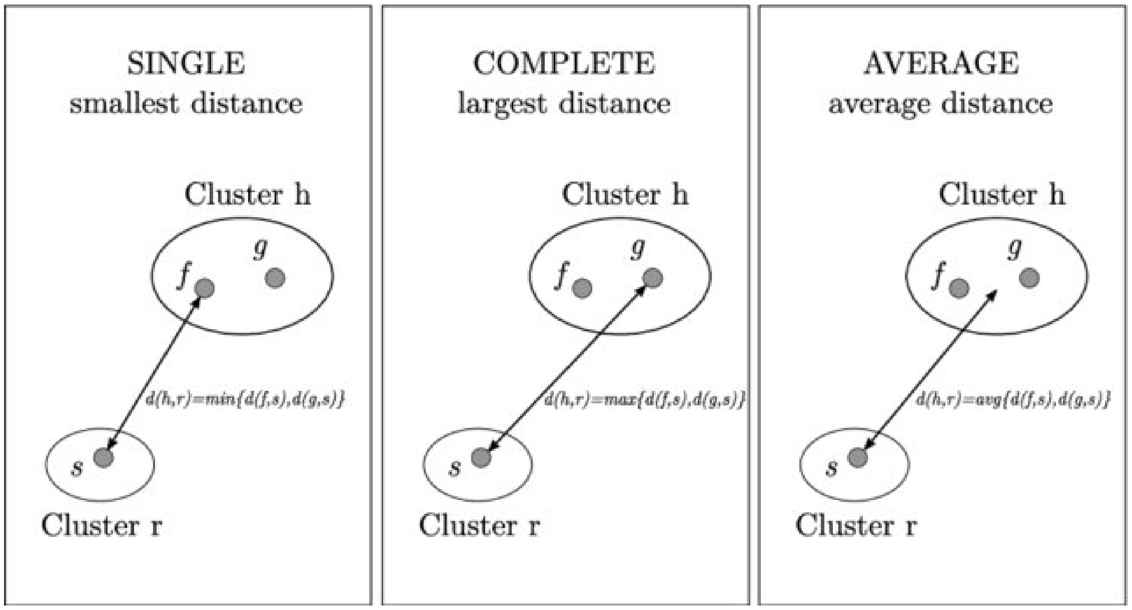
2. For $i = n, n-1, \ldots, 2$,

   (a) calculate all the inter-cluster dissimilarities and identify the pair of two clusters that are the least dissimilar
   (b) merge these two clusters where the dissimilarity indicates the height in the dendrogram at which the merge takes place
   (c) calculate the new inter-cluster dissimilarities among the remaining $i-1$ clusters

Dissimilarity between 2 groups

- Need to generalize the notion of dissimilarity between a pair of observations, to the dissimilarity between a pair of groups (inter-cluster).

- We use the notion of linkage that defines the dissimilarity between 2 groups of observations.



- Four types of linkage :

$\Rightarrow$ complete
$\Rightarrow$ average
$\Rightarrow$ single
$\Rightarrow$ centroid

- We usually choose complete or average, since they produce more balanced dendrograms.

- Centroid is often used in genomics.

Further indications

- For the dissimilarity between 2 observations, we use:
  - ⇒ Euclidean distance (most used)
  - ⇒ Correlation distance
- The choice of this distance is very important, and is context-dependent.
- Scaling: as for k-means, we must pay attention to the context.
- It is strongly recommended to test several choices of:
  - ⇒ distance
  - ⇒ linkage
  - ⇒ cut height

Example

- The function `hclust()` performs hierarchical clustering in R.

- Take the following simulated data :

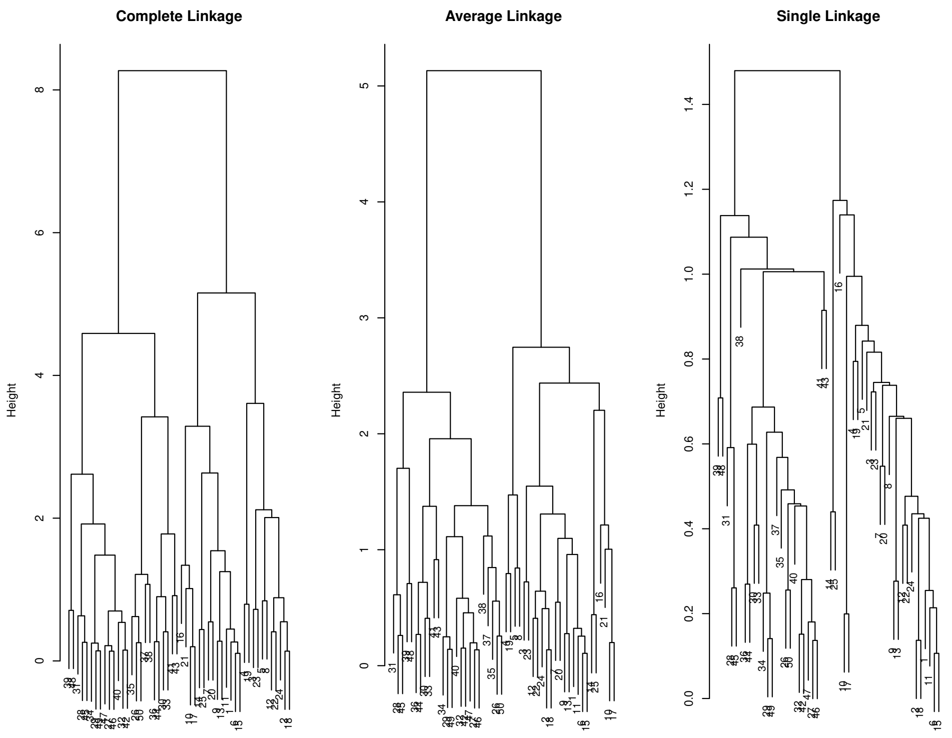
```
set.seed (222)
x=matrix (rnorm (50*2) , ncol =2)
x[1:25 ,1] = x[1:25 ,1] + 3
x[1:25 ,2] = x[1:25 ,2] - 4
head(x)
##          [,1]      [,2]
## [1,] 4.487757 -3.405585
## [2,] 2.998108 -3.479132
## [3,] 4.381021 -4.952044
## [4,] 2.619786 -5.227685
## [5,] 3.184136 -4.202407
## [6,] 2.753104 -2.940880
```

- Use the Euclidean distance to compute the clusterings with complete, average and single linkages.

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

- Plot the dendrograms.

```
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)
```



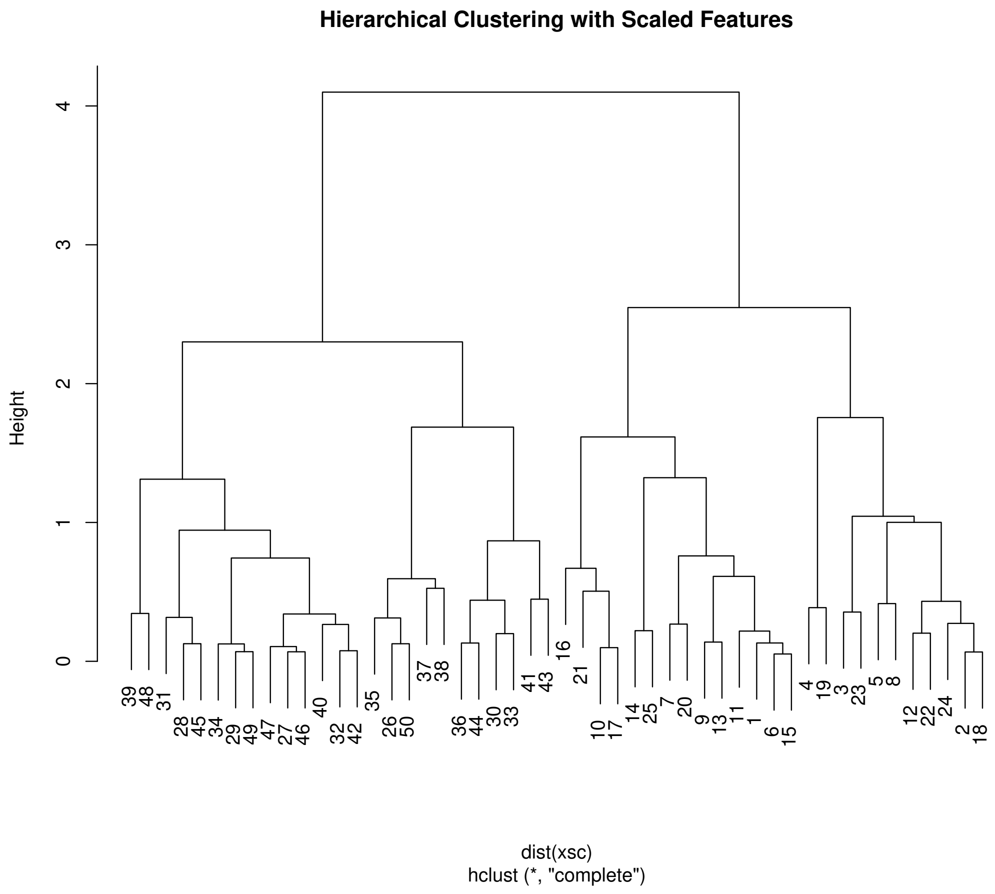
- Display the cluster labels for each observation associated with a given cut

```
cutree(hc.complete, 2)
[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 2 2 2
[30] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
cutree(hc.average, 2)
[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 2 2 2
[30] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
cutree(hc.single, 2)
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
[30] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

- The complete and average linkages give good results, but the simple is not accurate.

- For scaling, we can use the function `scale()`

```
xsc=scale(x)
plot(hclust(dist(xsc), method="complete"),
   main="Hierarchical Clustering with Scaled Features")
```


Hierarchical Clustering with Scaled Features
Height
0
1
2
3
4
39 48 31 28 45 34 29 49 47 27 46 40 32 42 35 26 50 37 38 36 44 30 33 41 43 16 21 10 17 14 25 7 20 9 13 11 1 6 15 4 19 3 23 5 8 12 22 24 2 18
dist(xsc)
hclust (*, "complete")

References

1. M. DeGroot, M. Schervish, *Probability and Statistics*, Addison Wesley, 2002.

2. Spiegel, Murray and Larry Stephens, *Schaum's Outline of Statistics,* 6th edition, McGraw Hill. 2017.

3. G. James, D. Witten, T. Hastie, R. Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer. 2013.

4. T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*. Springer. 2009.

5. Rachel Schutt and Cathy O'Neil. *Doing Data Science.* O'Reilly. 2014.