Outline: **Cluster Analysis**
(F. Chiaromonte)

Introduction to Statistical Learning
Chapter 10 section 3 and 4 Lab 2

Units\Features	X_1	X_2	…	X_p
Unit 1	x_{11}	x_{12}		x_{1p}
Unit 2	x_{21}	x_{22}		x_{2p}
. . .				
Unit n	x_{n1}	x_{n2}		x_{np}

p features measured on n units:

- An n x p data matrix X
- A data cloud of n points in R^p .

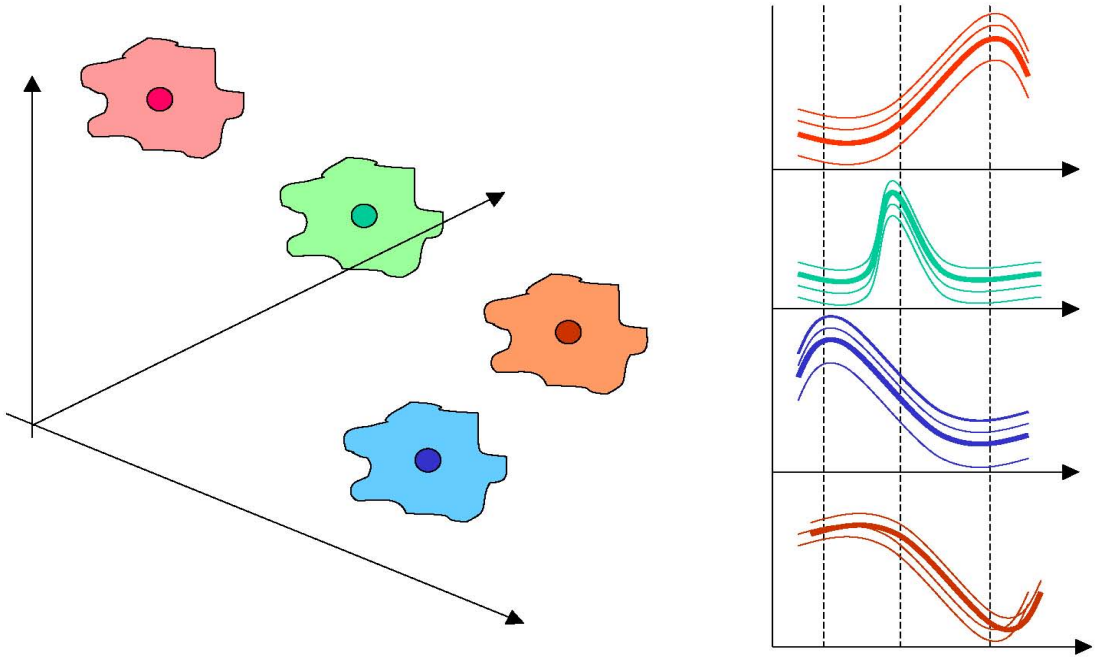
General idea: form groups of features (columns) or more commonly groups of units (rows) that are similar within groups and distinct across groups.

Unsupervised classification (define group labels on the basis of similarity).

"Discover" group structure in the data.

Simple methods determine a **hard partition**: every point belongs to one group, and one group only.

- **(Agglomerative) Hierarchical clustering**
- **K-means clustering**



Basic cartoon: partition and characteristic profiles

Pre-processing of the data matters, for instance:

• Center and/or scale by row/unit? Then a typical Euclidean distance will score as similar units with similar profiles across features, regardless of their location and scale. The same is achieved using a correlation distance.

• Center and/or scale by column/feature? (then a typical Euclidean distance will be insensitive to original differences in variability scales for different features)

(Agglomerative) Hierarchical Clustering

Set of n data points in $\boldsymbol{R}^p$.

• Choose a **distance function** for points $d(x_1,x_2)$ (Euclidean distance; correlation distance; other more complicated distances). Sometimes the point distance is not defined explicitly as a function, but provided through an nxn matrix.

• Choose a distance function for clusters, i.e. a **linkage function**, $D(C_1,C_2)$ based on a summary of the distances between points as measured by $d(x_1,x_2)$ (for clusters formed by just one points, D reduces to d).

Proceeding in an agglomerative fashion ("bottom-up"), generate a sequence of nested partitions of the data – progressively less fine:

- Start from n clusters, each containing one data point.
- At each iteration:

1. Using the current matrix of cluster distances, find the two closest clusters.
2. Update the list of clusters by merging the two closest.
3. Update the matrix of cluster distances accordingly

- Repeat until all data points are joined in one cluster.

Remarks:

- The method is sensitive to anomalous data points/outliers
- Mergers are irreversible: "bad" mergers occurring early on affect the structure of the nested sequence (path dependence).
- If two pairs of clusters are equally (and maximally) close at a given iteration, we have to choose arbitrarily; the choice will affect the structure of the nested sequence.

Defining cluster distance: the linkage function

$$D(C_1, C_2) = \min_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

Single (string-like, long)

$$D(C_1, C_2) = \frac{1}{\#(C_1)\#(C_2)} \sum_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

Average

$$D(C_1, C_2) = \max_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2)$$

Complete (ball-like, compact)

$$D(C_1, C_2) = d(\bar{x}_1, \bar{x}_2)$$

Centroid

Single and complete linkages produce nested sequences invariant under monotone transformations of *d* – not the case for average linkage. However, the latter is a compromise between "long", "stringy" clusters produced by single, and "round", "compact" clusters produced by complete. Centroid linkage can cause inversions.

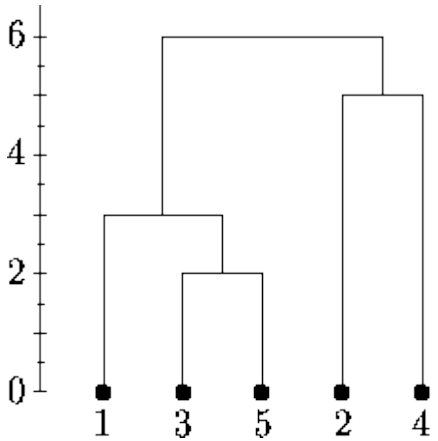
Example

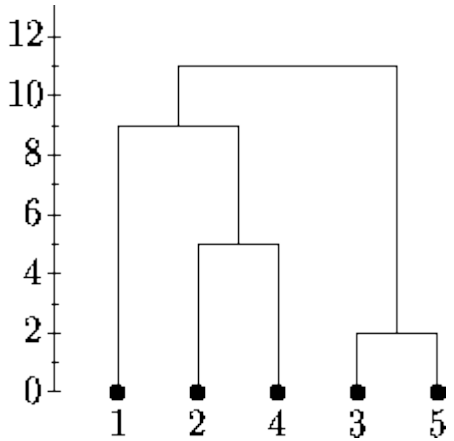
Agglomeration step in constructing the nested sequence (first iteration): Left: matrix of distances among 5 data points. 3 and 5 are the closest, and are therefore merged in cluster "35". Right: new distance matrix computed with complete linkage.

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	2	8	0

	35	1	2	4
35	0			
1	11	0		
2	10	9	0	
4	9	6	5	0

Dendrogram representing the nested sequence produced by single linkage (left) and complete linkage (right). Ordinate: distance, or hight, at which each merger occurred. Horizontal ordering of the data points is any order preventing intersections of branches.





IMPORTANT: hierarchical clustering, per se, does not dictate a partition and a number of clusters; it provides a nested sequence of partitions (this is more informative than just one partition). To settle on one partition, we have to **"cut" the dendrogram**.

Loosely: read the hight associated with various numbers of clusters as a measure of "fit" of the partitioning in K groups (more later).

K-means Clustering (a partitioning algorithm).

Set of n data points in $\boldsymbol{R}^p$.

- Choose a distance function for points $d(x_1,x_2)$.
- Choose K = number of clusters.
- Initialize the algorithm; two options
    - Initialize the K cluster centroids (e.g. K data points chosen at random)
    - Initialize the labels (e.g. each data point gets a random label from {1,2…K}), then compute centroids $\bar{x}_1(0),\ldots\bar{x}_K(0)$

Use the data to iteratively relocate centroids, and reallocate points to closest centroid.

- At each iteration:

1. Compute distance of each data point from each current centroid
$$d(x,\bar{x}_k) \quad \forall x,\ k=1\ldots K$$
2. Update current cluster membership of each data point, selecting the centroid to which the point is closest
$$m(x)=\arg\min_{k=1\ldots K} d(x,\bar{x}_k) \quad \forall x$$
3. Update current centroids, as averages of the new clusters formed in (2).
$$\bar{x}_k=\frac{1}{\#(x:m(x)=k)}\sum_{x:m(x)=k} x \quad k=1\ldots K$$

- Repeat until cluster memberships, and thus centroids, stop changing.

Remarks:

• Also this method is sensitive to anomalous data points/outliers.
• Points can move from one cluster to another, but the final solution depends strongly on initialization (another form of path dependence). ***Run algorithm on multiple initializations*** and select solution with best outcome (smallest total within cluster sum of squares).
• If two centroids are equally (and maximally) close to an observation at a given iteration, we have to choose arbitrarily; the problem here is not so serious because points can move.
• There are several "variants" of the K-means algorithm.

Objective function

K-means converges to a local minimum of the **total within cluster sum of squares** – not necessarily a global one; will depend on initialization.

$$W(K,start)=\sum_{k=1}^{K}\sum_{x:m(x)=k} d^2(x,\bar{x}_k) \propto \sum_{k=1}^{K}\sum_{x,\tilde{x}:m(x)=k} d^2(x,\tilde{x})$$

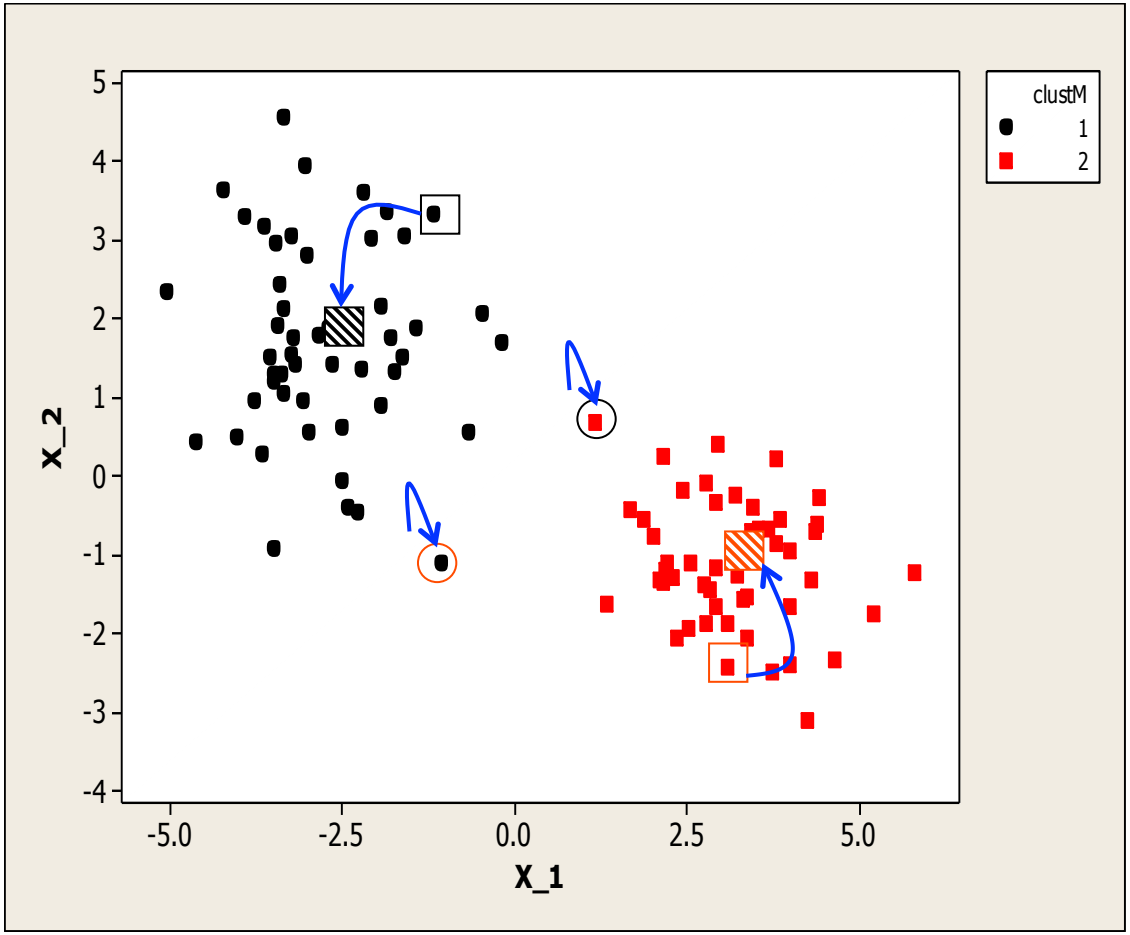
Clusters tend to be ball-shaped with respect to the chosen distance.

Example

Iteration step (first) in updating centroids and memberships.

Rectangles: random centroid initializations, and updated centroids.

Circles: updated memberships.



IMPORTANT: K-means creates only one partition with the specified number of clusters; to get partitions with different K's need to run the algorithm again (and nesting is not guaranteed).

Loosely: read the final sum of squares value associated with various numbers of clusters as a measure of "fit" of the partitioning in K groups (more later).

Variants on partitioning algorithms:

• Partitioning around medioids (PAM): instead of averages, use multivariate medians as centroids (cluster "prototypes"). Dudoit and Freedland (2002).

• Self-organizing maps (SOM): add an underlying "*topology*" (neighboring structure on a lattice) that relates cluster centroids to one another. Kohonen (1997), Tamayo et al. (1999).

• Fuzzy k-means: allow for a "gradation" of points between clusters; *soft partitions*. Gash and Eisen (2002).

• Mixture-based clustering: implemented through an EM (Expectation-Maximization) algorithm. This provides *soft partitioning*, and allows for *modeling* of cluster centroids and shapes. Yeung et al. (2001), McLachlan et al. (2002)

Evaluating a clustering solution:

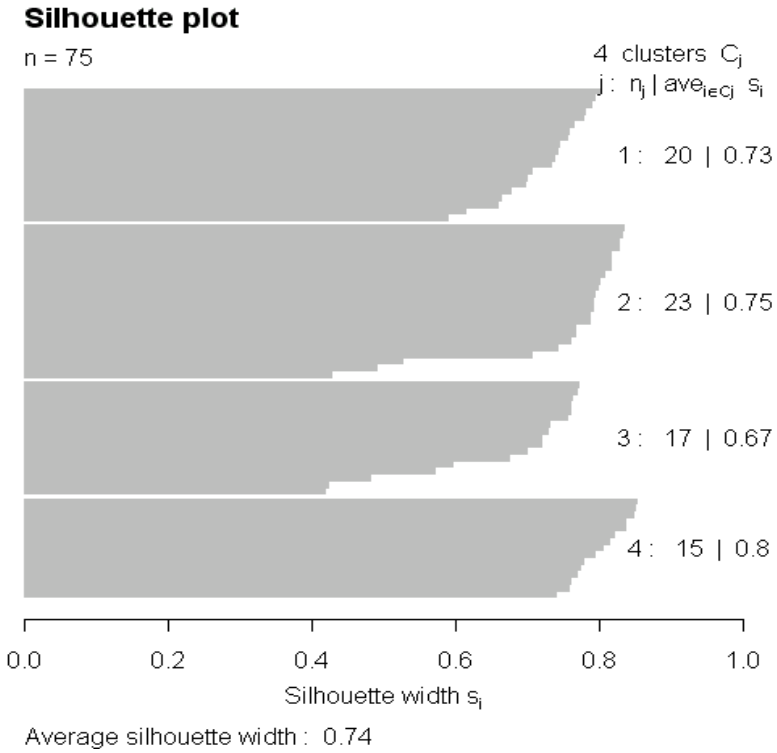
Besides dendrogram cut hight, or final value of the total within cluster sum of squares, a clustering can be evaluated trough **Silhouette widths**. For each point i=1…n define:

$$\delta(i,C) = \frac{1}{\#(C)} \sum_{j \in C} d(x_i, x_j)$$

$$a_i = \delta(i, C(i)) \quad C(i) = C \text{ st } i \in C$$

$$b_i = \min_{C \neq C(i)} \delta(i,C)$$

$$sil_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$



Silhouette of 75 units in 4 clusters:
Cluster sizes and average silhouette widths:

20	23	17	15
0.7262347	0.7548344	0.6691154	0.8042285

Individual silhouette widths:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.4196	0.7145	0.7642	0.7377	0.7984	0.8549

Observations with large and positive *sil* (~1) are well clustered, those with *sil* around 0 lie between clusters, and those with negative *sil* are placed in the "wrong" cluster.

Summary statistics for sil_i , i=1…N; plots of them (most commonly, by cluster).

Computational assessments:

Because methods are so sensitive to possibly "anomalous" positions of points, and distributional assumptions hard to make, computational assessments (stability, bootstrapping) are also very important:

- Perturb *adding noise* (drawn from what distribution?) to the data.

- Perturb *deleting points* (resample without replacement) from the data.

- Bootstrap (resample with replacement) the data.

Is the clustering solution (dendrogram and chosen partition; partition in K groups and choice of K) stable to perturbations? What is their sampling variability?

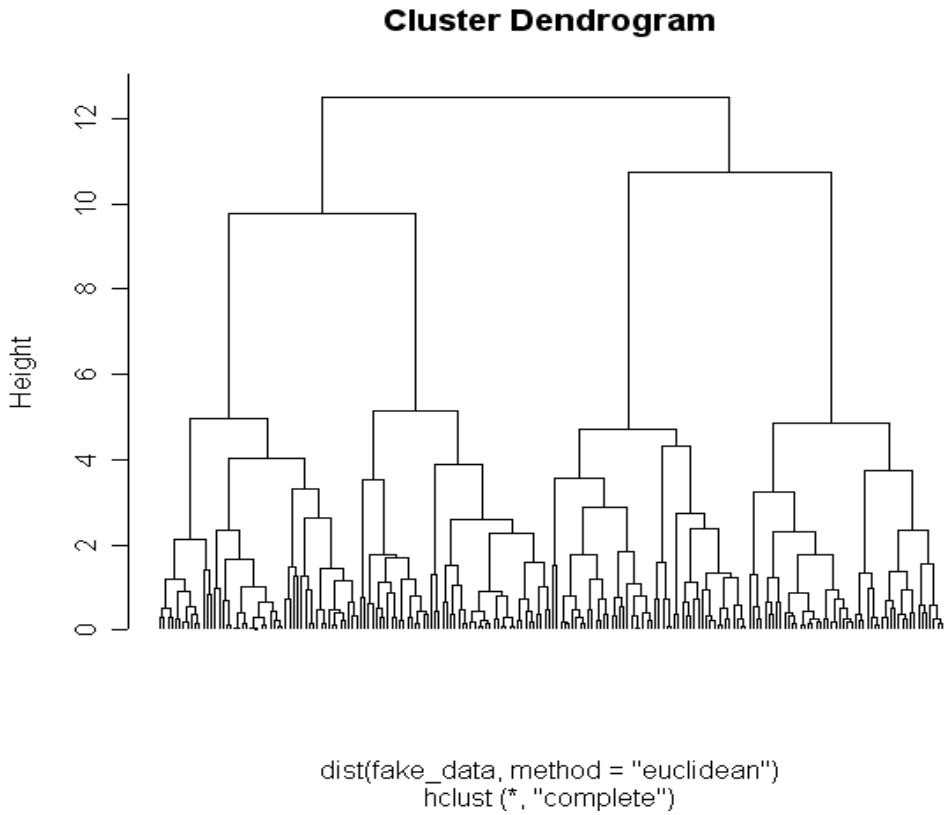
Approaches to determine the number of clusters in a data set

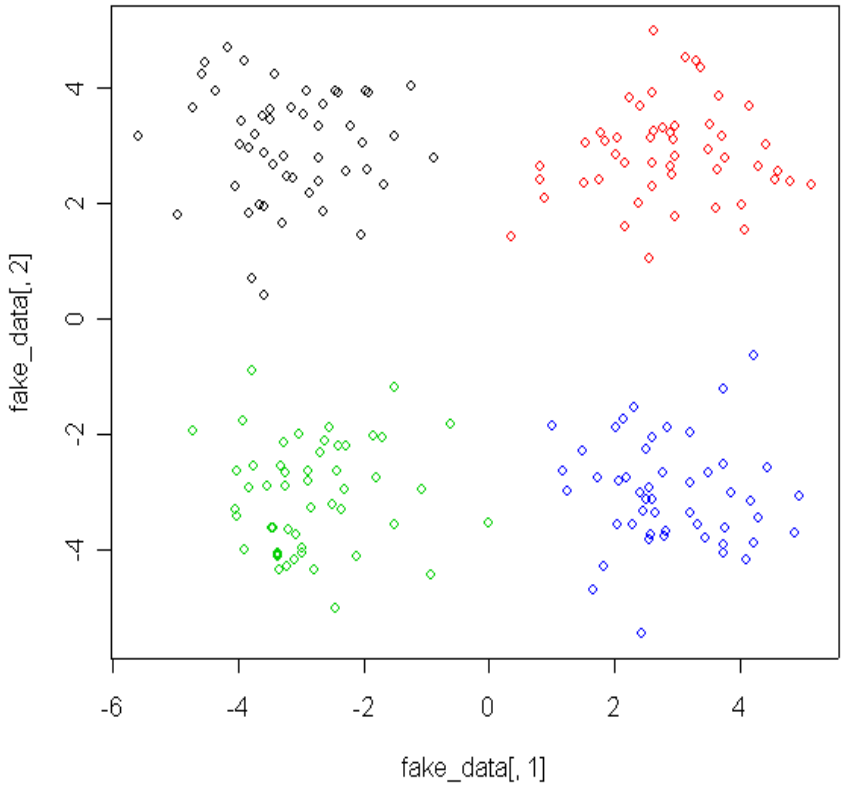
Data set: x_i , i=1…n points in $\boldsymbol{R}^p$ (each coordinate is a feature for the clustering)

Clustering method: e.g. hierarchical with given choices of distance (e.g. Euclidean) and linkage (e.g. complete); K-means with given choice of distance (e.g. Euclidean).

With given method and K (# clusters), we obtain a partition of the points: $P(K) = \{C_1 \ldots C_K\}$

For instance, fake 2D data set (n=200, mixture of four $N(\mu_j, I_2)$)





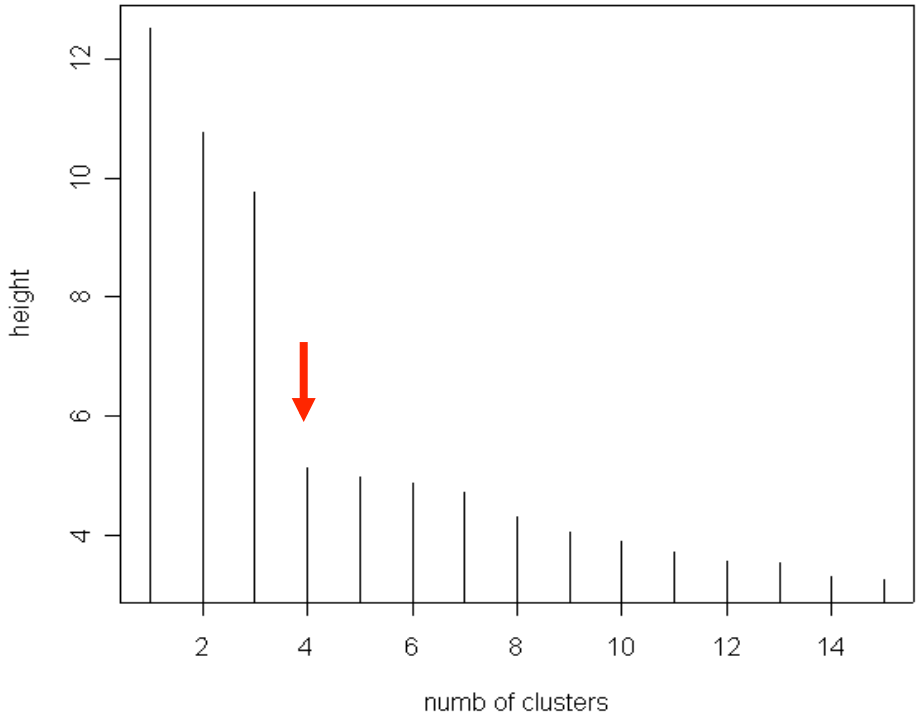
(Hierarchical and K-means give identical results here)

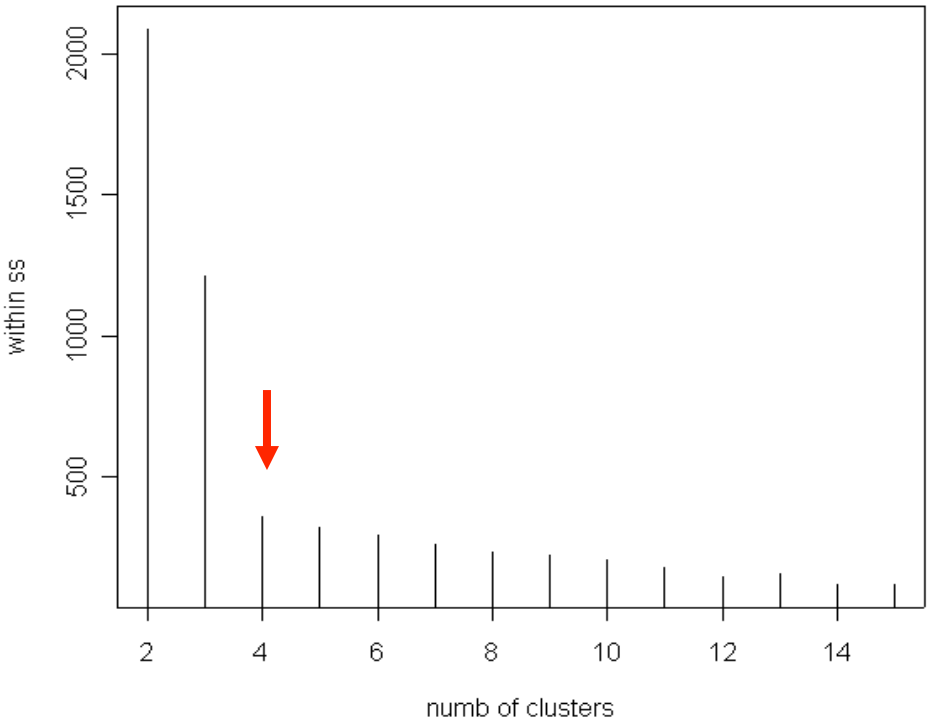
Within cluster dissimilarity/distance

Hierarchical: Dissimilarity levels (heights) at which clusters are formed by cutting.

K-means: Within clusters sum of squares (what the algorithm finds a local minimum for)

$$W(K) = \sum_{j=1\ldots K} \sum_{i \in C_j} d^2(x_i; \bar{x}_j)$$



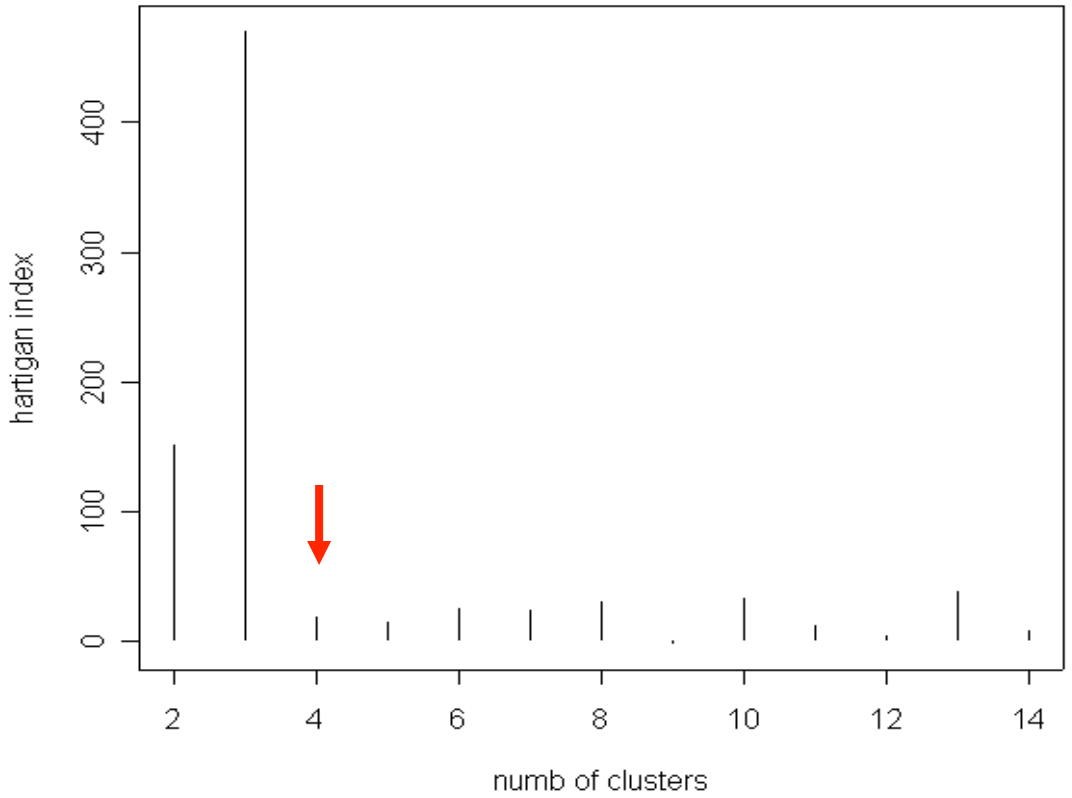


Low values when the partition is good, BUT these are by construction monotone non-increasing (more clusters always makes $\leq$ within cluster dissimilarity); look for "bends".

Hartigan Index

$$H(K) = \gamma(K)\frac{W(K) - W(K+1)}{W(K+1)}$$

correction $\gamma(K) = n - K - 1$



(corrected) relative improvement when passing from K to K +1. High value (right before) followed by low value when the partition is good. NOT monotone.

Average Silhouette

$$d_{i,C} = \frac{1}{\#(C)} \sum_{l \in C} d(x_i, x_l)$$

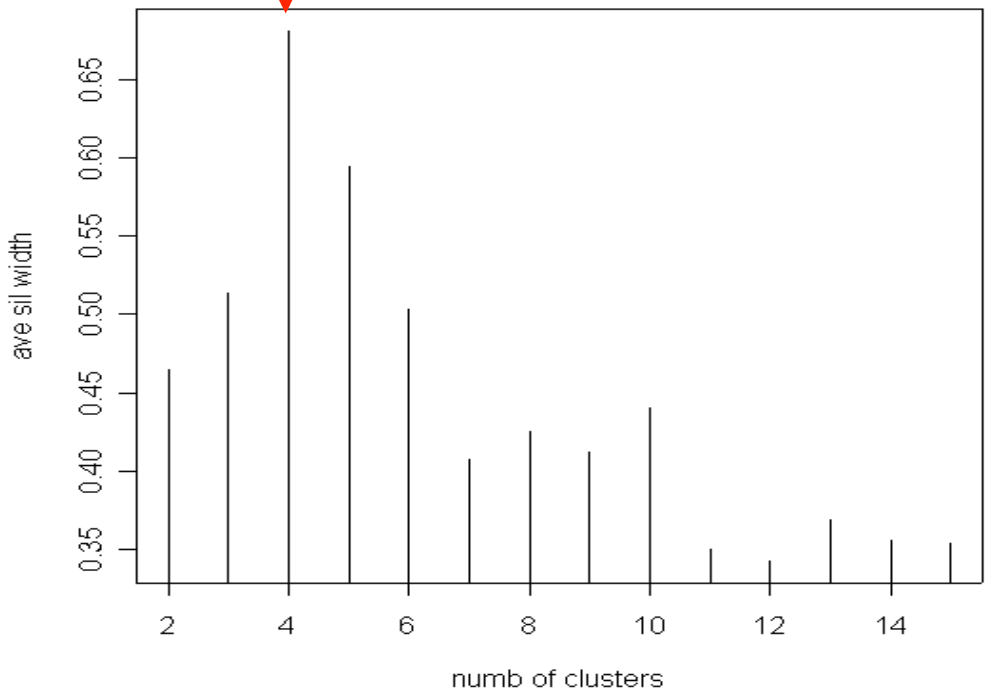
$$a_i = d_{i,C(i)} \qquad b_i = \min_{C \neq C(i)} d_{i,C}$$

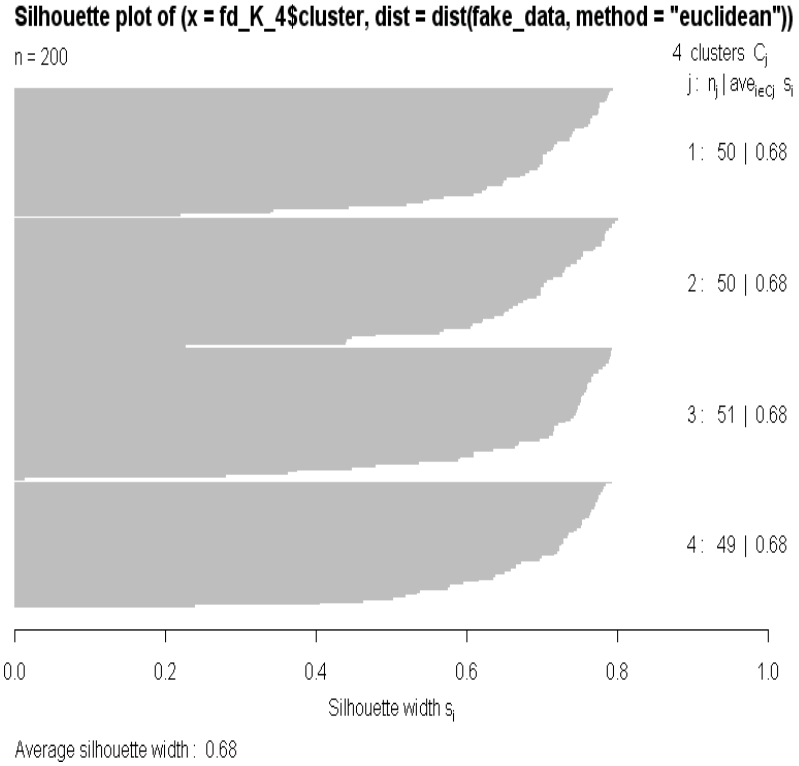
$$Sil_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

How well a point is clustered

$$Sil(K) = \frac{1}{N} \sum_{i=1\ldots N} sil_i$$

Averaging over points, overall partition quality





High value when the partition is good. NOT monotone.

Remarks:

- Many other approaches are possible, e.g., for each *K*, benchmark one of these quality indexes based on stability or reproducibility of the clustering solution.

- Also, for Mixture-based clustering, Bayes Information Criterion (BIC) can be computed for each *K* and used to select a *K*.

- In many real applications the "right" number of clusters is not at all evident; the data points do not show distinct groups – clustering is still useful; think of it more like a way to segment a data cloud in (not well separated) subparts.