Non-hierarchical clustering

$$\sum_{i=1}^{n}\sum_{i'=1}^{n} d(i,i') = \underbrace{\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(i,i')}_{D_{within}} + \underbrace{\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d(i,i')}_{D_{between}}$$

Too many ways to cluster n observations in K groups
↳ use algorithm → K-mean

K-means is developed for quantitative variables and it uses the squared Euclidean distance as measure of dissimilarity

$$D_{within} = \sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} \|x_i - x_{i'}\|^2 = 2\sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - m_k\|^2$$

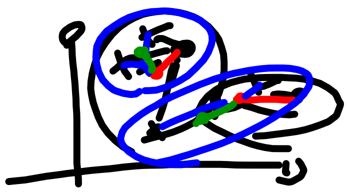
where:
n_k = size of cluster k (number of observations in the k-th cluster);
m_k = mean vector of the observations in the k-th cluster

The method aims at minimizing D_{within} given a number of clusters K and initial values for m_k. It proceeds iteratively by allocating the observations to the clusters and updating the m_k's

K-means

1) Choose K and (arbitrary) values for m_k's
2) Repeat
   a) for $i=1,\dots,n$, assign x_i to the group k, $k = \arg\min_c \|x_i - m_c\|^2$
   b) for $k=1,\dots,K$ compute m_k as the arithmetic mean of all the observations of group k

   until the m_k's do not change anymore



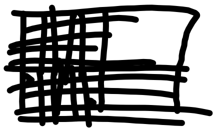
Note:
2a assures that D_{within} is minimum given m_k
2b " " D_{within} " " once the observation are assigned to the group k

- In the example, K=3. If we set K=4, the algorithm find an extra group, even if "it is not there". Often is not a huge problem, as we can consider two groups as one.
- k-means is not able to catch "filiform" type of clustering due to the use of the Euclidean distance → sometimes different dissimilarities are more useful
- in the example, the algorithm found the same clusters when using different starting points: often the case in simple examples, but actually it is not guaranteed (k-means may converge to a local minimum)

LIMITATIONS
- it depends on initial (arbitrary) choices (K, m_i's)
- it can be applied only to quantitative variables

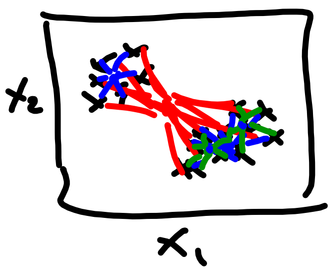
The latter can be solved by using different dissimilarities
→ from centroids (means) to medoids

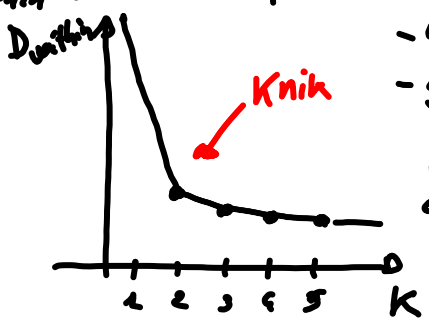
To mitigate the first issue:
- re-run the algorithm from different starting points (m_i's)
- implement heuristic rules to find the best K (Gap statistics and similar ...)

Choice of K


- increasing K, D_{within} gets always smaller
- the particularity is that even if we evaluate the clustering in a test set, we will get smaller D_{within} by increasing K (finer partitioning of the space)

The general idea is that if we create new clusters when they are actually there, the is a huge drop in D_{within}, while if we create a new cluster when the is actually no need, then the decrease in D_{within} will be really small





- quite heuristic
- gap statistic: it contrast $\log D_{within}$ with the same quantity computed on data uniformly distributed in the space