

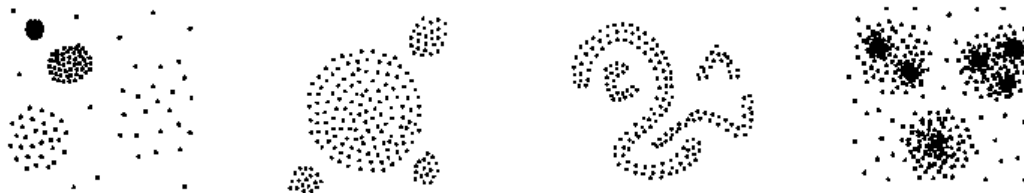


# Clustering

Programming for Data Science

## Goal of clustering

- Partition a set of objects/data into groups –so called clusters— so that objects in the same cluster are as similar as possible and objects in different clusters are as dissimilar as possible.



## Properties of clusters

- Clusters can vary in size, shape, and density
- Disjunct vs. overlapping
- Deterministic vs. probabilistic
- Hierarchical vs. flat

# Clustering by Minimization of Variance

## *Algorithm characteristics*

- converges against a (possibly only local) minimum
- complexity:  $O(n)$  for one iteration
- Number of iterations is typically small (~5-10)

## *Further hard to quantify properties*

- Sensitivity to noise / outliers
- Convex cluster shape
- Result and runtime strongly depend on initialization (first random segmentation)

## Variant of variance minimizing algorithm

- Centroids are re-calculated whenever an object changes cluster affiliation
- Two main Versions: Lloyd & MacQueen
- **Lloyds Algorithm:** see next slide
- **MacQueen's Algorithm:**  
Incremental adaption, when a point  $p$  changes from Cluster  $C_1$  to Cluster  $C_2$

$$\bar{x}_j(C'_1) = \frac{1}{n_{C_1}-1} \cdot (n_{C_1} \cdot \bar{x}_j(C_1) - x_j^p) \quad \text{und} \quad \bar{x}_j(C'_2) = \frac{1}{n_{C_2}+1} \cdot (n_{C_2} \cdot \bar{x}_j(C_2) + x_j^p)$$

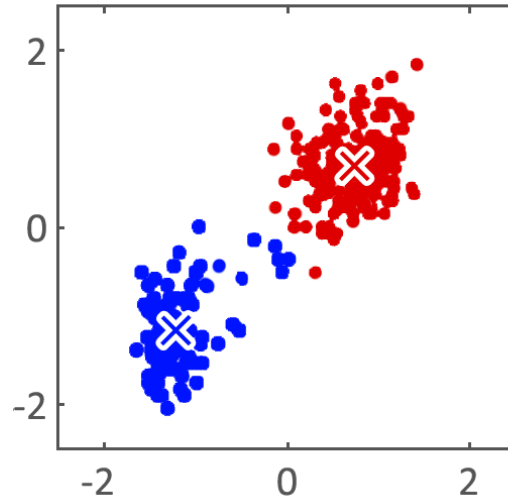
- $n_C$  = number of objects in  $C$

## Remarks

- Most common clustering method (by far)
- complexity:  $O(nkt)$ ,  $n$ : #objects,  $k$ : #clusters,  $t$ : #iterations, typically:  $k \ll n$  and  $t \ll n$
- k-means is basically variance minimizing, and strongly depends on the processing order

# kMeans - Lloyd's Algorithm

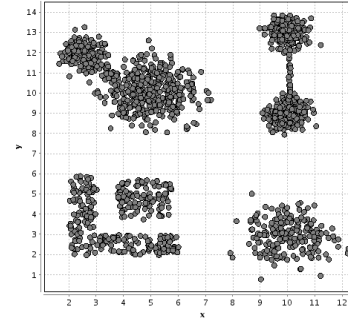
```
ClusteringByVarianceMinimization(data set, k)  
Create random initial k cluster centroids  $C'=\{C_1, \dots, C_k\}$ ;  
 $C = \{\}$ ;  
repeat until  $C = C'$   
     $C = C'$ ;  
    Create k clusters, by assigning each point to its nearest centroid;  
    Calculate  $C'=\{C'_1, \dots, C'_k\}$  for the newly defined clusters.  
return C;
```



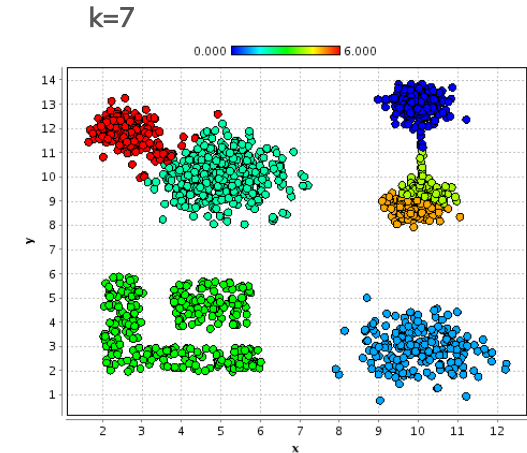
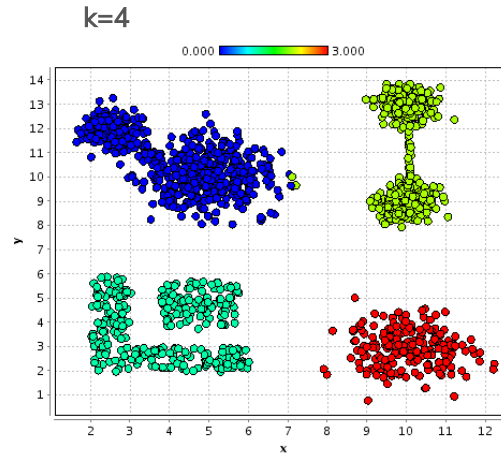
# K-Means

## Example data

- $d=2$  (x-y-coordinates)
- Synthetically generated



## K-Means with different $k$



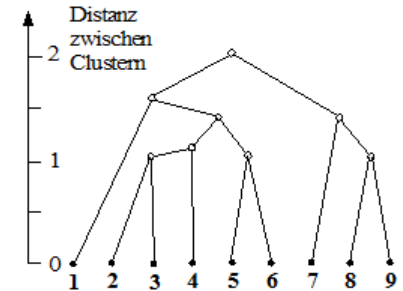
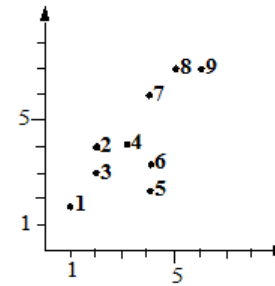
# Hierarchical clustering

## Goal

- Create a representation of the data, from which clusters can be derived

## Dendrogram

- Merge clusters with minimal distance
- Tree, whose nodes represent clusters
  - Root contains whole data set
  - Leaves contain individual objects
  - Inner node is union of all objects contained in its child nodes



Dendrogram

## Types

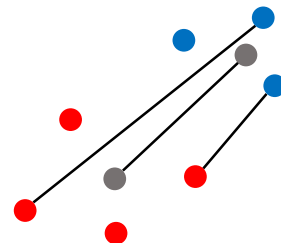
- Bottom-Up
  - Agglomerative
- Top-Down
  - Divisive

## Algorithm

1. Create initial clusters, one object per cluster, and calculate the distance between all cluster pairs
2. Create new cluster by merging the two closest clusters
3. Calculate distances between new and old clusters
4. Stop if one cluster contains all objects: otherwise repeat from 2.

## Distance functions for object sets (cluster)

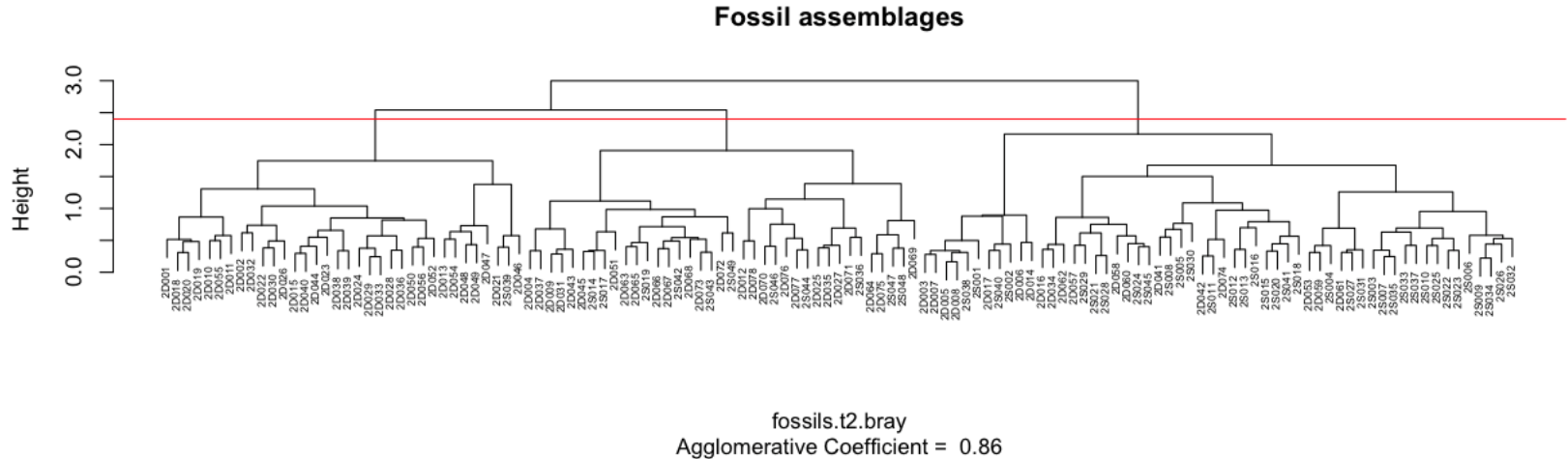
- Single-Link:  $dist.sl(X, Y) = \min_{x \in X, y \in Y} dist(x, y)$
- Complete-Link:  $dist.cl(X, Y) = \max_{x \in X, y \in Y} dist(x, y)$
- Average-Link:  $dist.al(X, Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} dist(x, y)$
- Centroid-Link:  $dist.ctl(X, Y) = dist(c_x, c_y)$





# Agglomerative Hierarchical Clustering

## Cutting out clusters



# Agglomerative Hierarchical Clustering

*Build the dendrogram „bottom-up“*

## Pros

- No knowledge on  $k$
- Identifies hierarchies, not only flat clusters
- A clustering is derived from the dendrogram with an horizontal cut

## Cons

- Sensitive to noise (Single-Link)
- Two clusters cannot be identified in the dendrogram if they are connected by a “line” of objects with similar inter-object distances
- Inefficient: runtime complexity  $O(kn^2)$  for  $n$  objects and  $k$  levels
- No incremental update

# Task

## Step 0

- You will get a csv file from us. Load it in your language/environment.
- Explore the data in it.

## Step 1

- Implement kMeans (Lloyd)\*.

## Step 2

- Implement AGNES\*. Choose at least one link criteria.

\*use your own implementation

# Package suggestions

R

- (data.table)

python3

- numpy
- pandas

# Exercise Appointment

## *We compare and discuss the results*

- Tuesday, 26.11.2019,
- Consultation: 21.11.2019,
- Please prepare your solutions! Send us your code!

*If you have questions, please mail us:*

[claudio.hartmann@tu-dresden.de](mailto:claudio.hartmann@tu-dresden.de) Orga + Code

[lucas.woltmann@tu-dresden.de](mailto:lucas.woltmann@tu-dresden.de) Tasks + Python

[lars.kegel@tu-dresden.de](mailto:lars.kegel@tu-dresden.de) Tasks + R

